

Exploiting Uncertainty Quantification in Derivative-Free Optimization

SIAM Conference on Computational Science and Engineering

Stephen Billups Jeffrey Larson ¹

University of Colorado Denver

March 1, 2011

¹Supported by NSF Grant GK-12-0742434

Outline

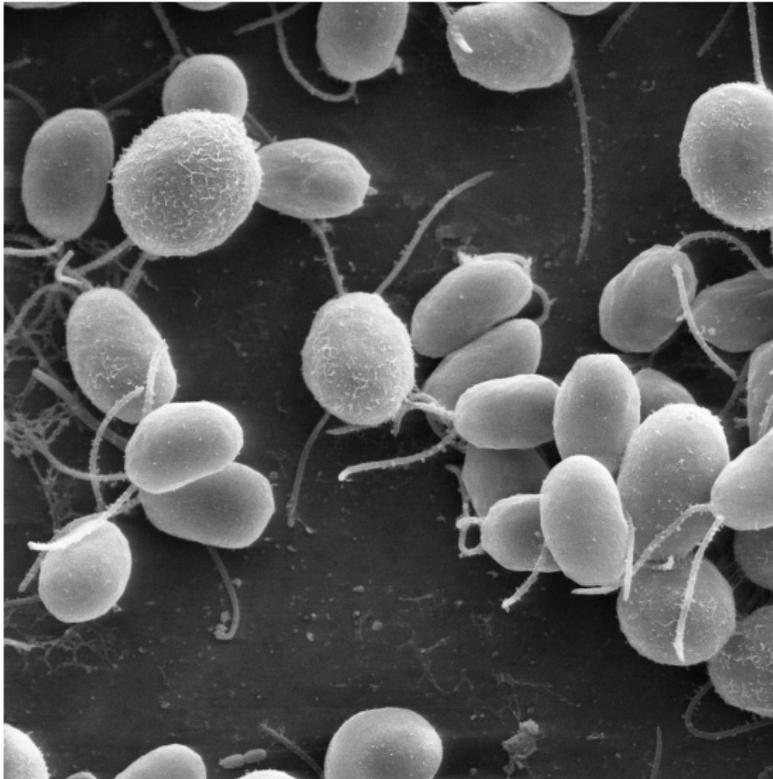
- 1 Overview
 - Why Derivative Free?
 - Example
- 2 Trust Region Methods
 - CSV Framework
 - Extra Care
- 3 Exploiting Uncertainty
 - Easy Problems
 - Harder Problems

Why No Derivatives?

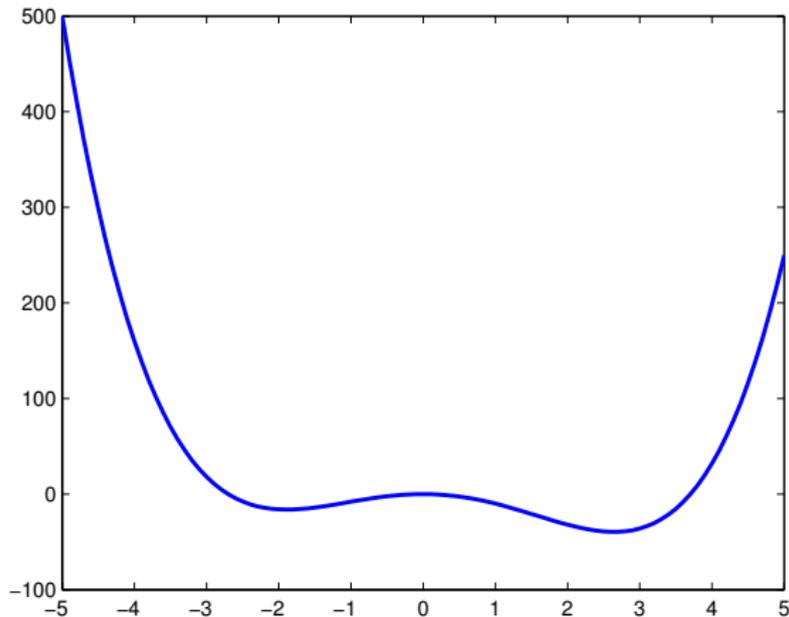
Sometimes just finding the function value is hard enough:

- Expensive function evaluations
 - Simulations involving nonlinear PDE's
- Unsupported legacy code which returns only function values
 - Finite differences requires N or $2N$ function calls
- Black-box functions unsuitable for automatic differentiation
 - Code involving "switch" statements for discrete parameters

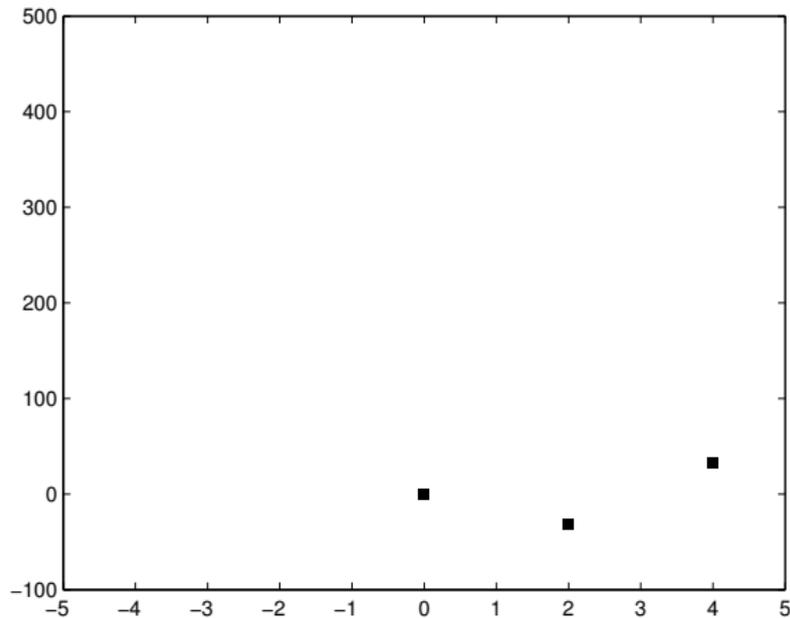
Application



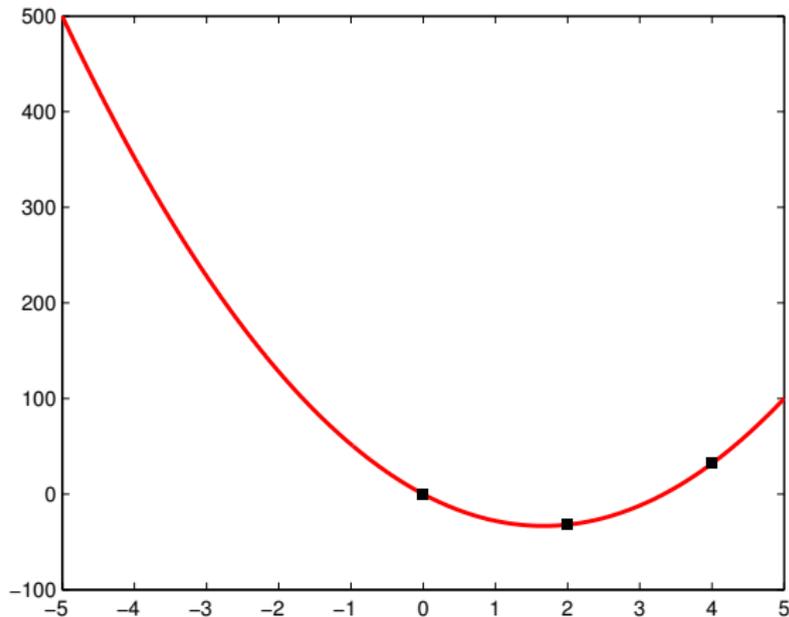
Interpolation Models



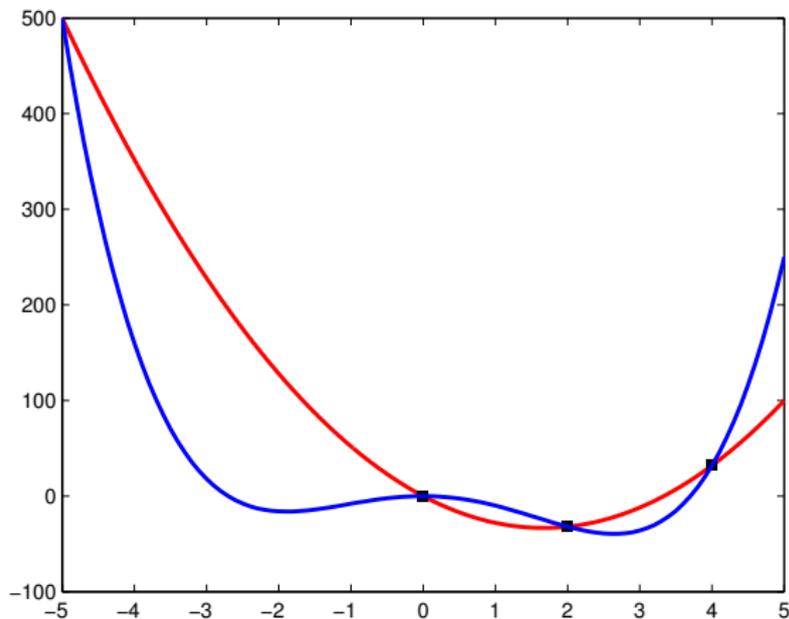
Interpolation Models



Interpolation Models



Interpolation Models



CSV Framework - Quadratic Models

Given a function f to minimize, an initial starting point x_0 , and trust-region radius Δ_0 , build a model m_0 :

- 1 Criticality Step
- 2 Compute a step s_k which minimizes $m_k(x_k + s_k)$ on $B(x_k, \Delta_k)$
- 3 Compute $f(x_k + s_k)$ and $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$
- 4 Update Δ_k and m_k

CSV Framework - Quadratic Models

Given a function f to minimize, an initial starting point x_0 , and trust-region radius Δ_0 , build a model m_0 :

- 1 Criticality Step
- 2 Compute a step s_k which minimizes $m_k(x_k + s_k)$ on $B(x_k, \Delta_k)$
- 3 Compute $f(x_k + s_k)$ and $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$
- 4 Update Δ_k and m_k

CSV Framework - Quadratic Models

Given a function f to minimize, an initial starting point x_0 , and trust-region radius Δ_0 , build a model m_0 :

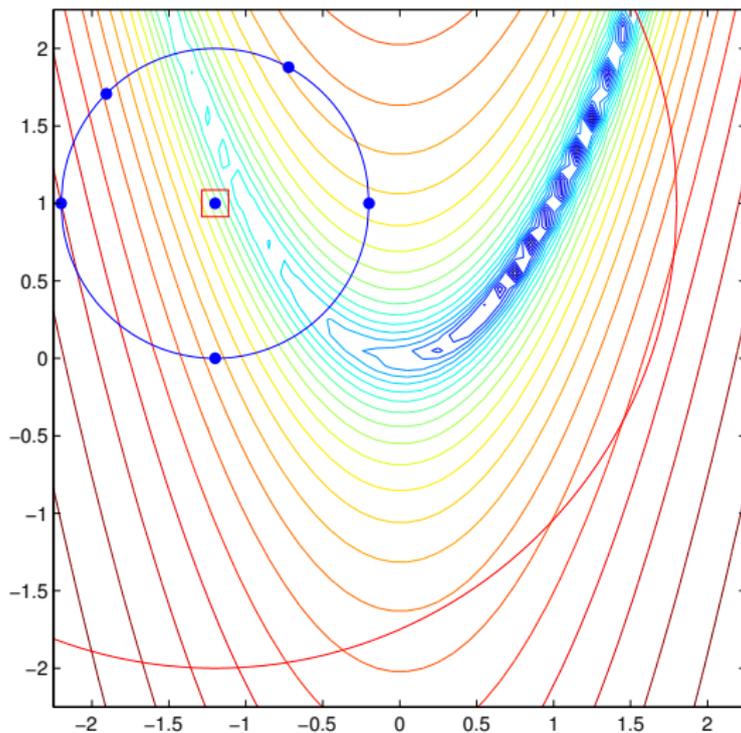
- 1 Criticality Step
- 2 Compute a step s_k which minimizes $m_k(x_k + s_k)$ on $B(x_k, \Delta_k)$
- 3 Compute $f(x_k + s_k)$ and $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$
- 4 Update Δ_k and m_k

CSV Framework - Quadratic Models

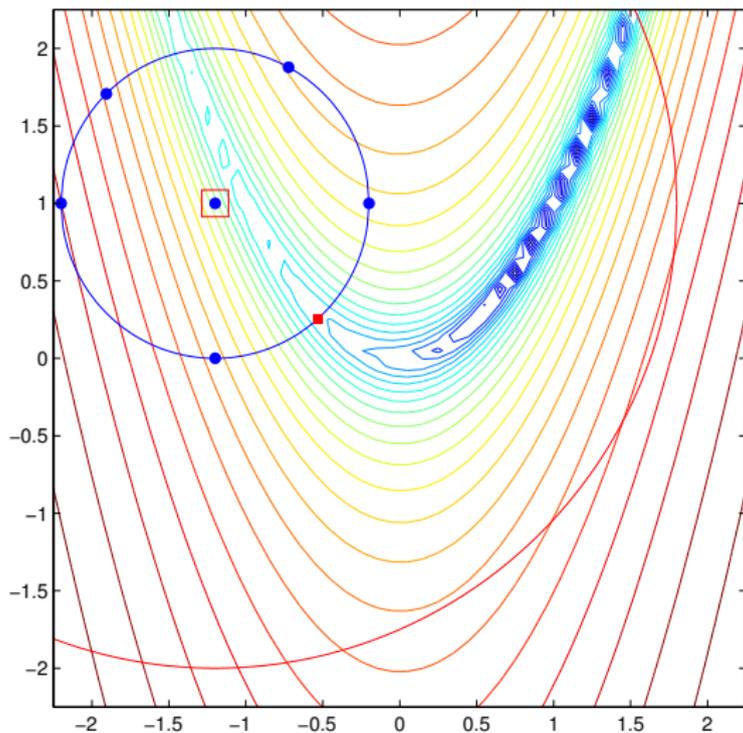
Given a function f to minimize, an initial starting point x_0 , and trust-region radius Δ_0 , build a model m_0 :

- ① Criticality Step
- ② Compute a step s_k which minimizes $m_k(x_k + s_k)$ on $B(x_k, \Delta_k)$
- ③ Compute $f(x_k + s_k)$ and $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$
- ④ Update Δ_k and m_k

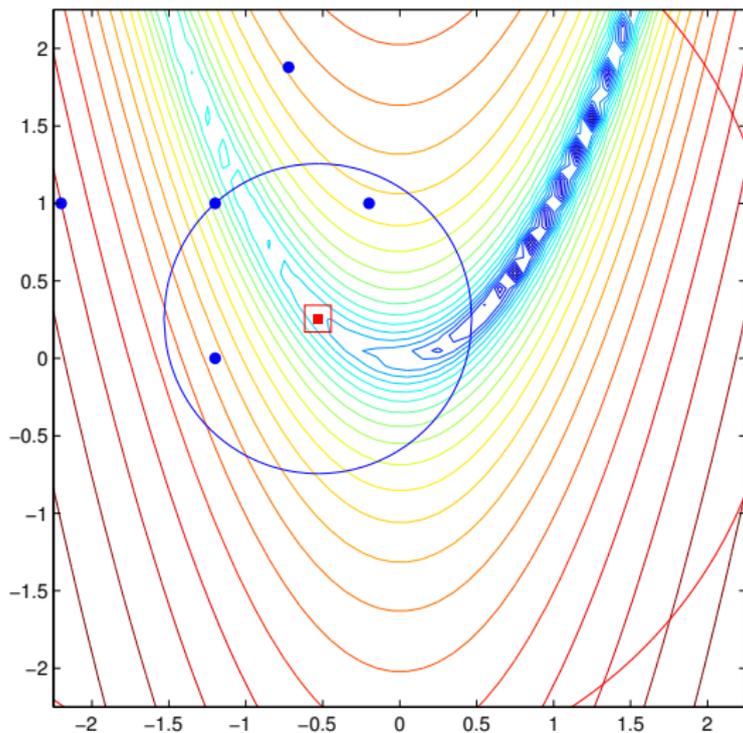
CSV - Interpolation



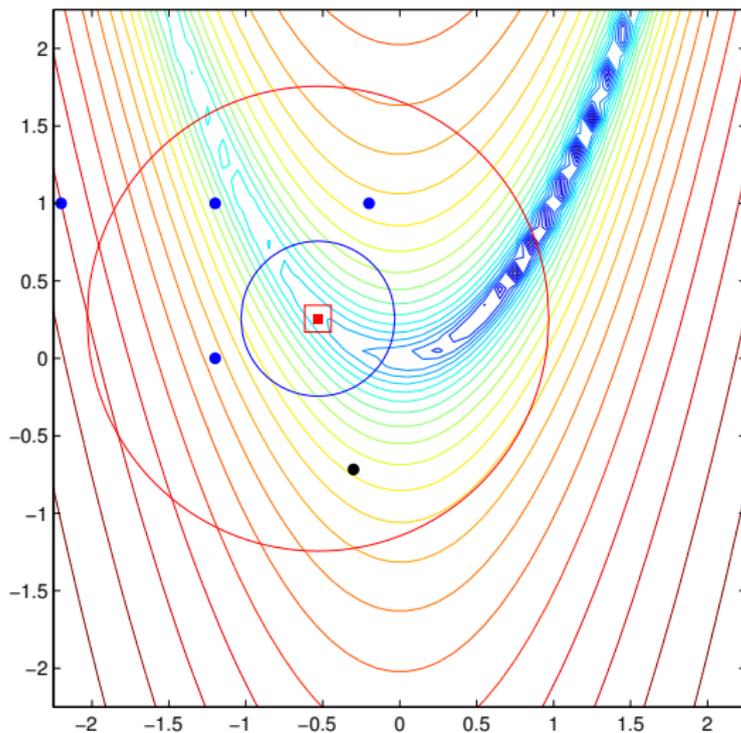
CSV - Interpolation



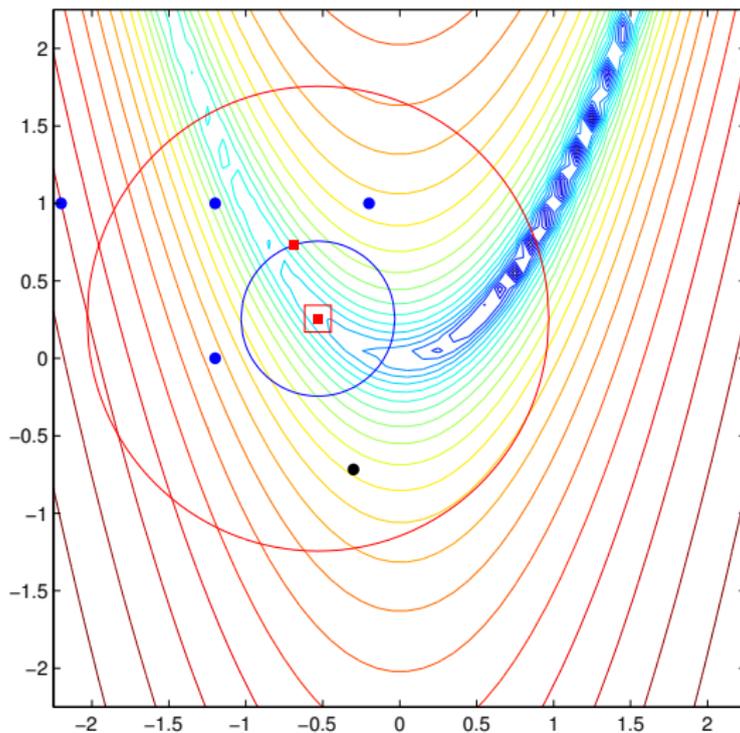
CSV - Interpolation



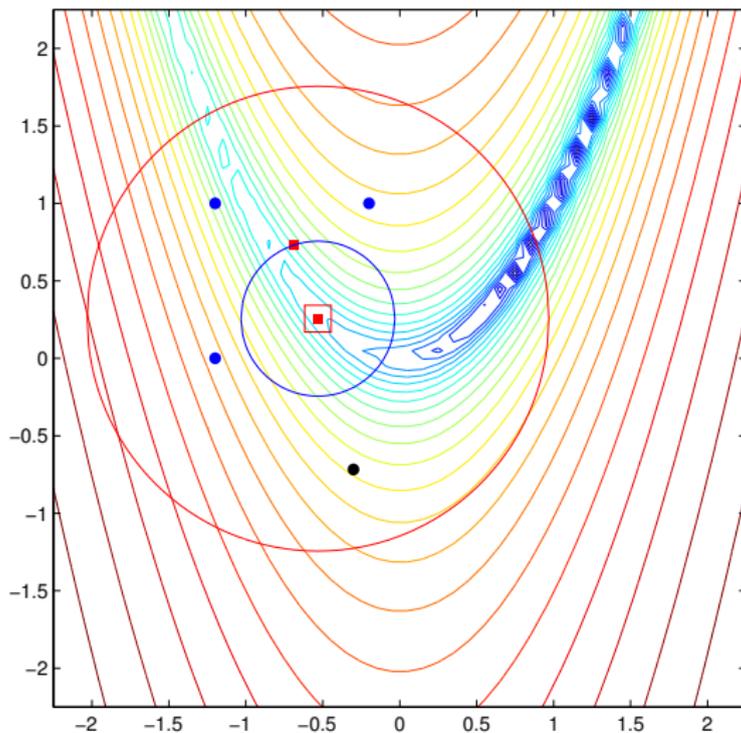
CSV - Interpolation



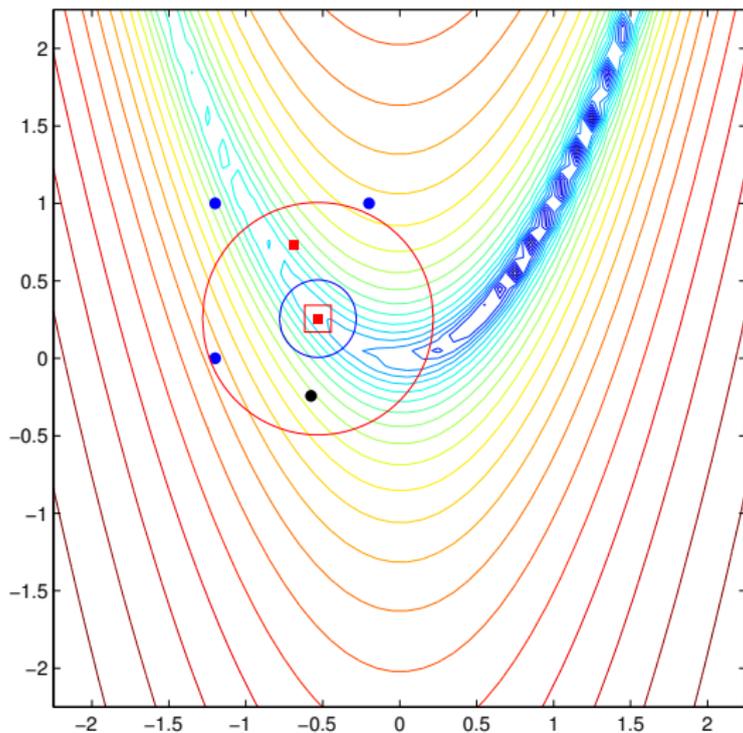
CSV - Interpolation



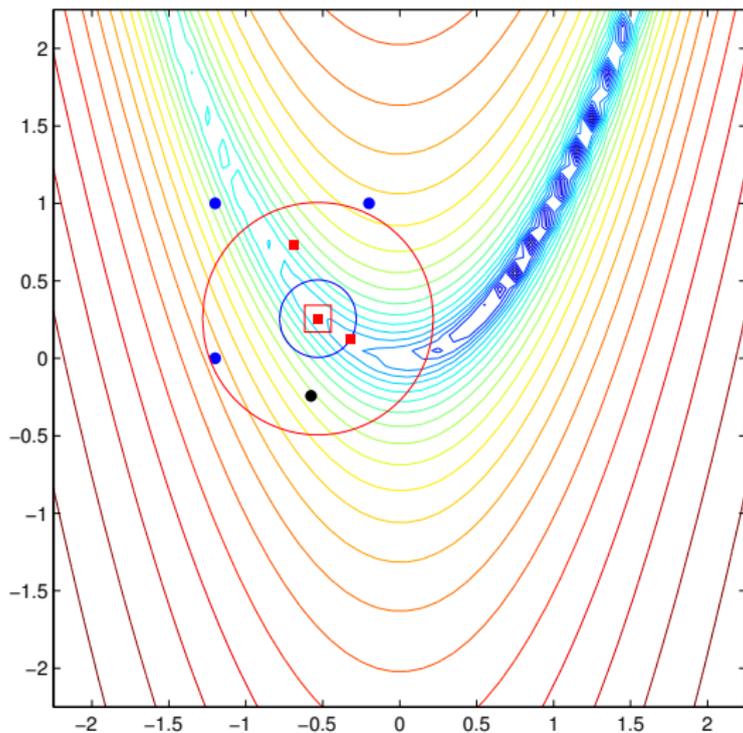
CSV - Interpolation



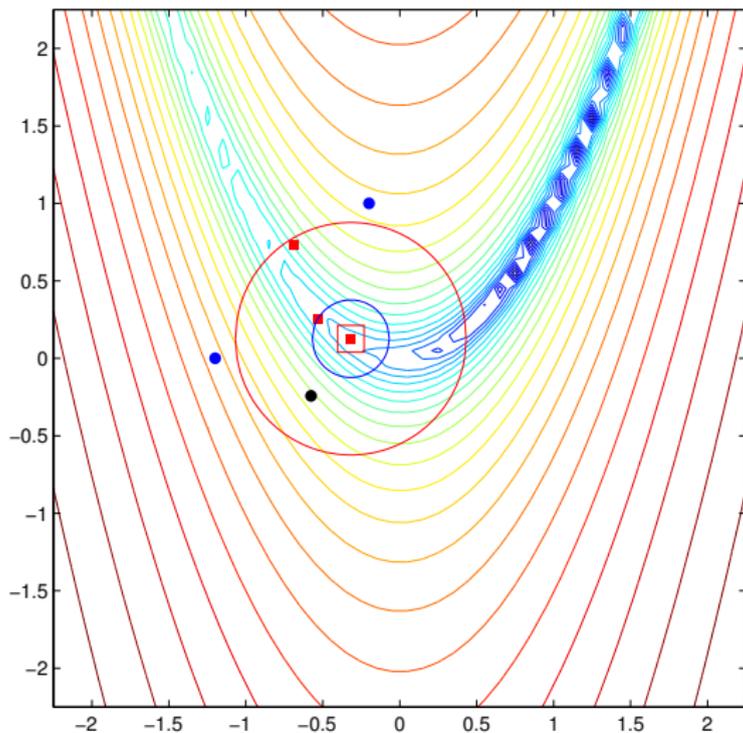
CSV - Interpolation



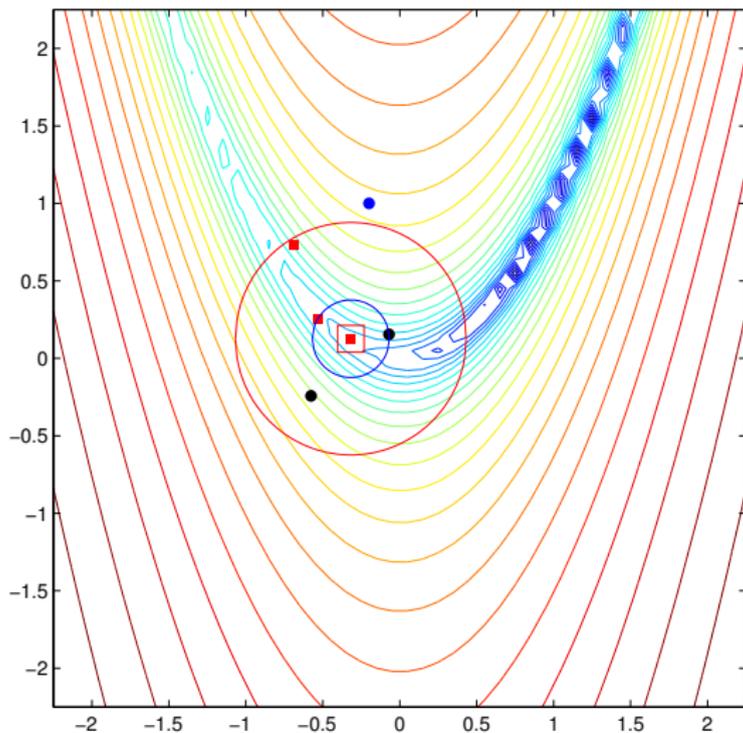
CSV - Interpolation



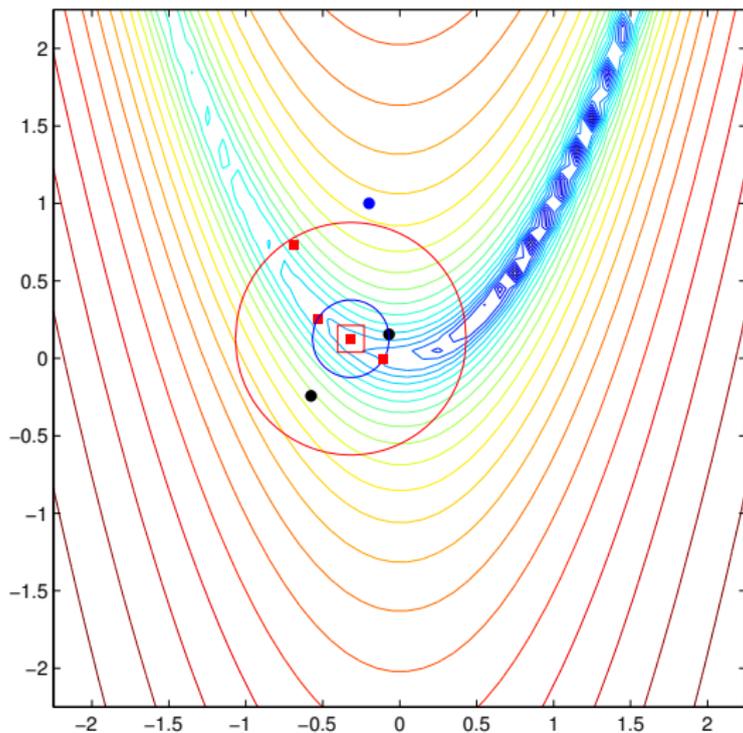
CSV - Interpolation



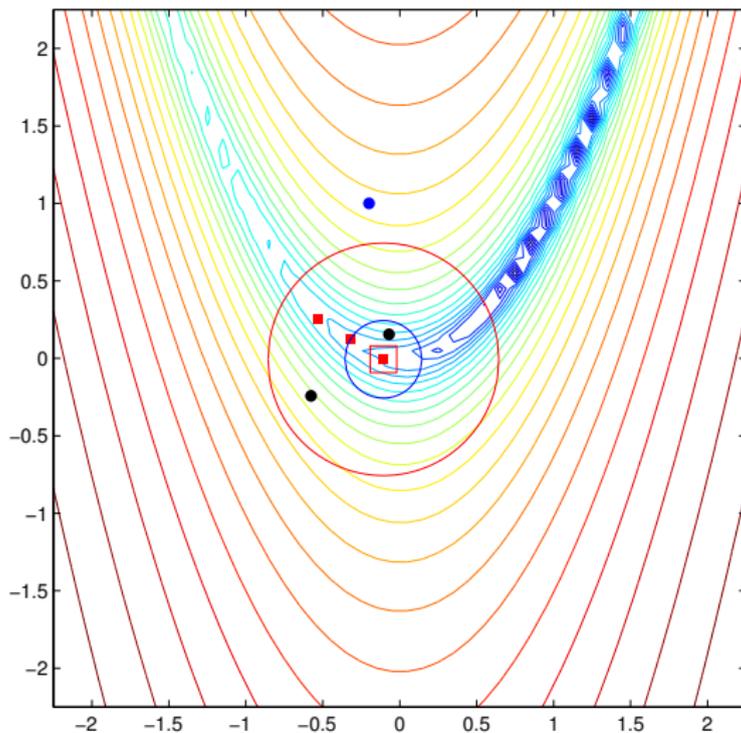
CSV - Interpolation



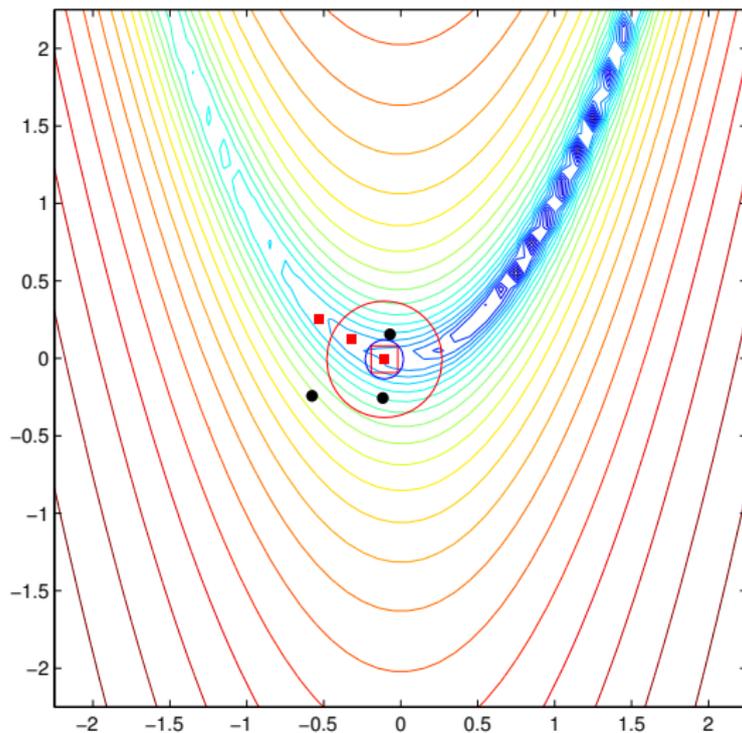
CSV - Interpolation



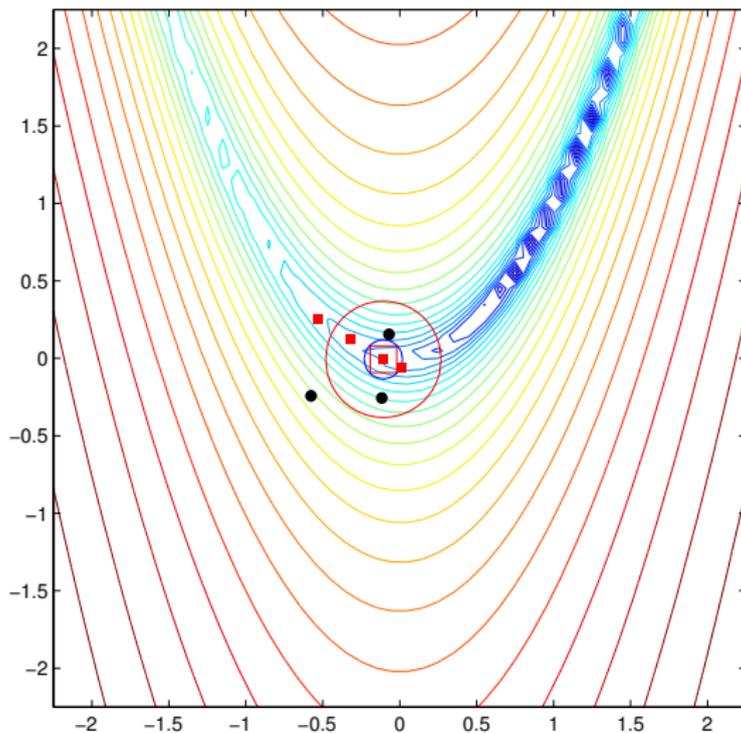
CSV - Interpolation



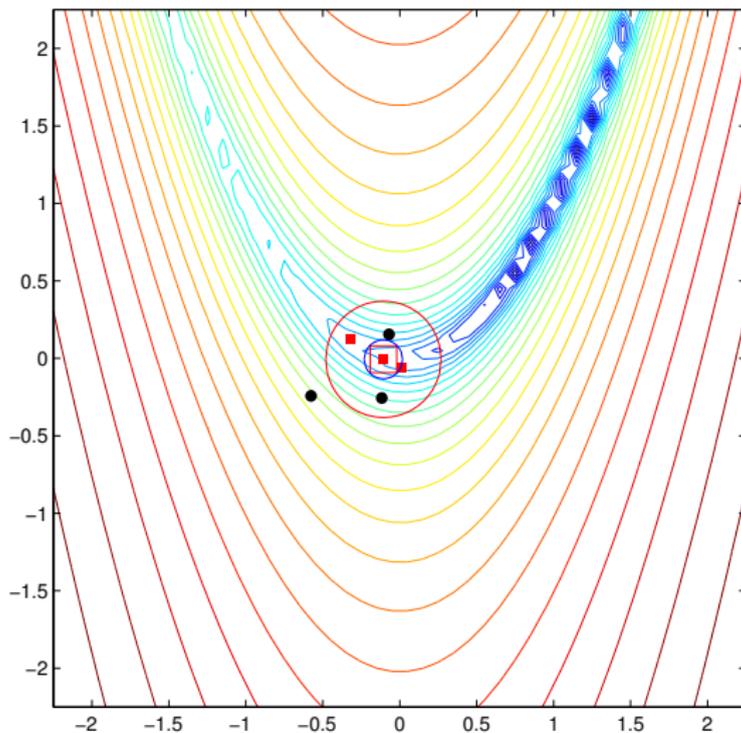
CSV - Interpolation



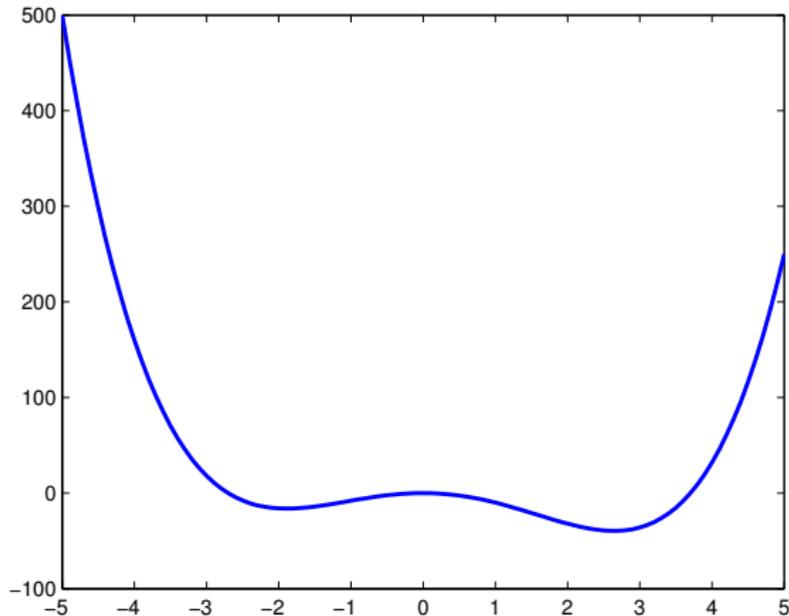
CSV - Interpolation



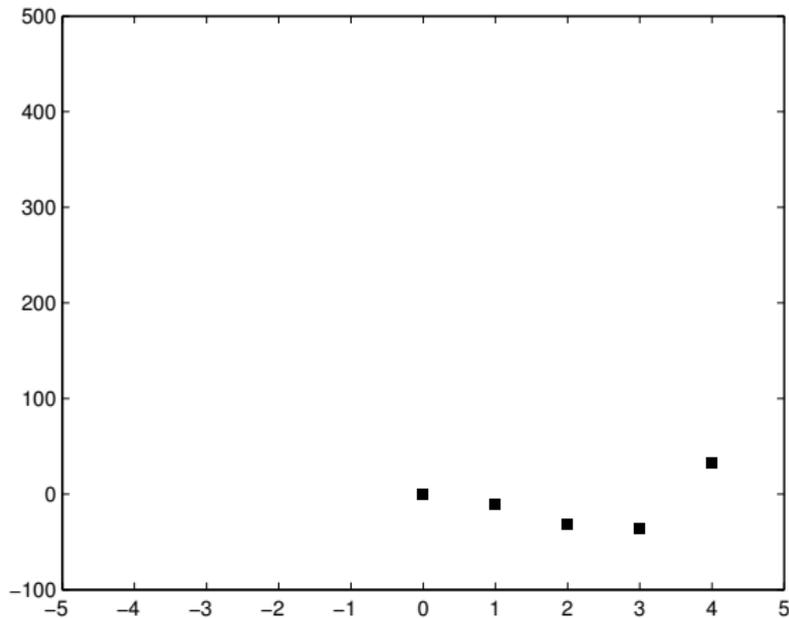
CSV - Interpolation



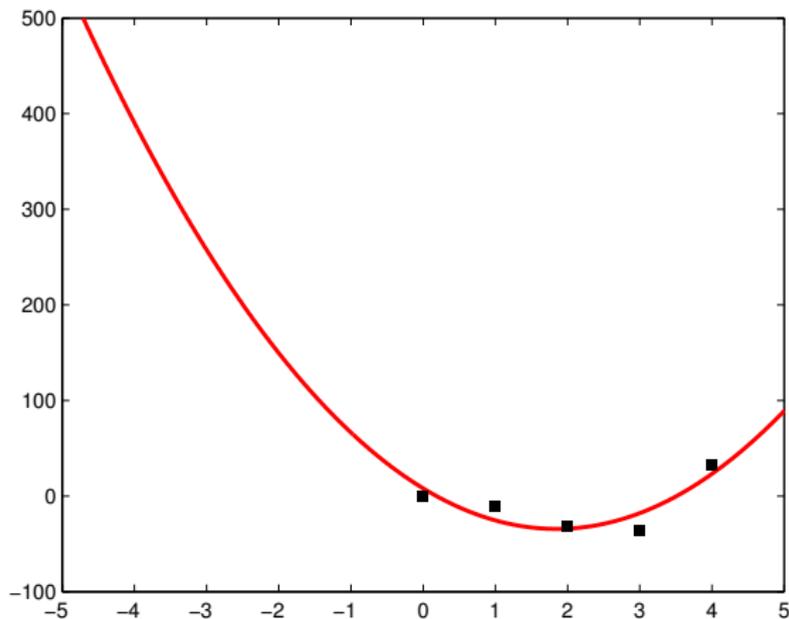
Regression Models



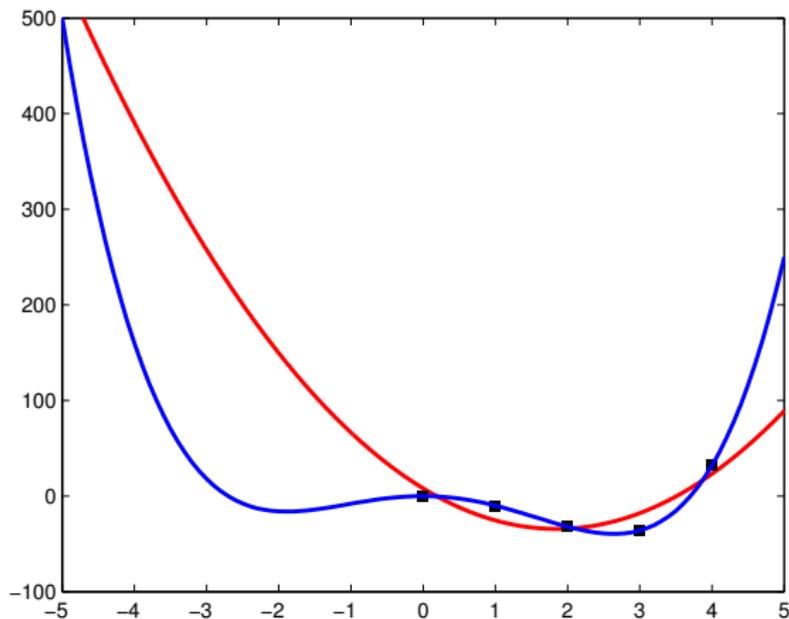
Regression Models



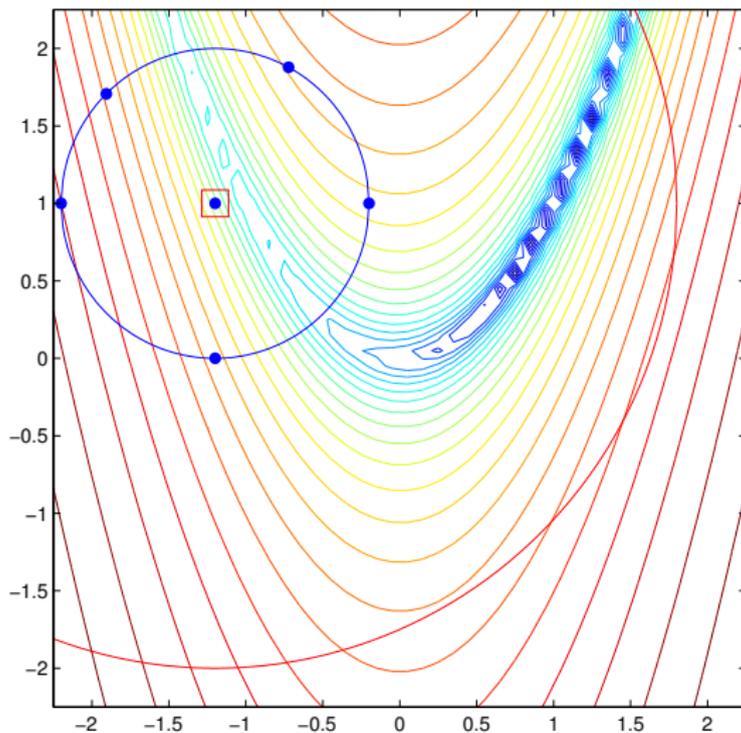
Regression Models



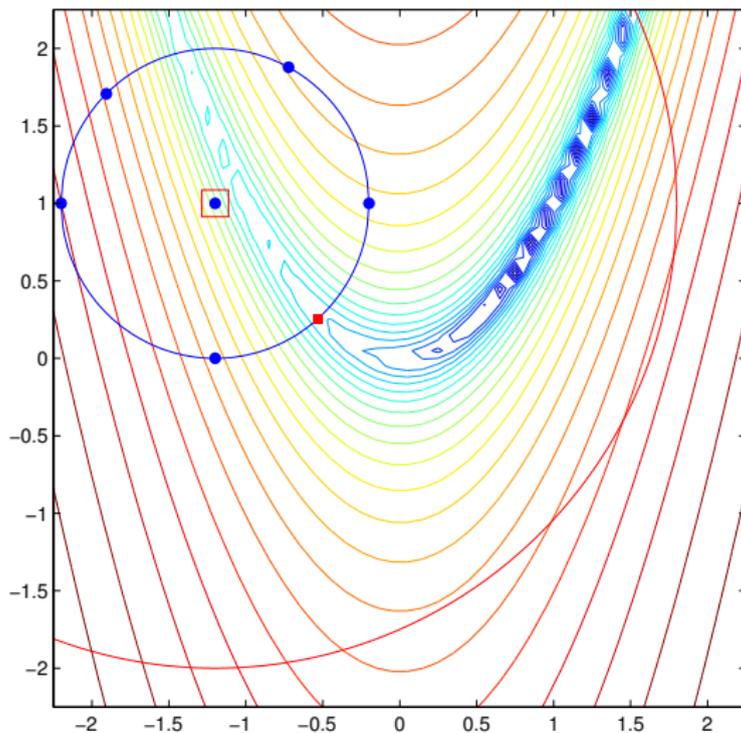
Regression Models



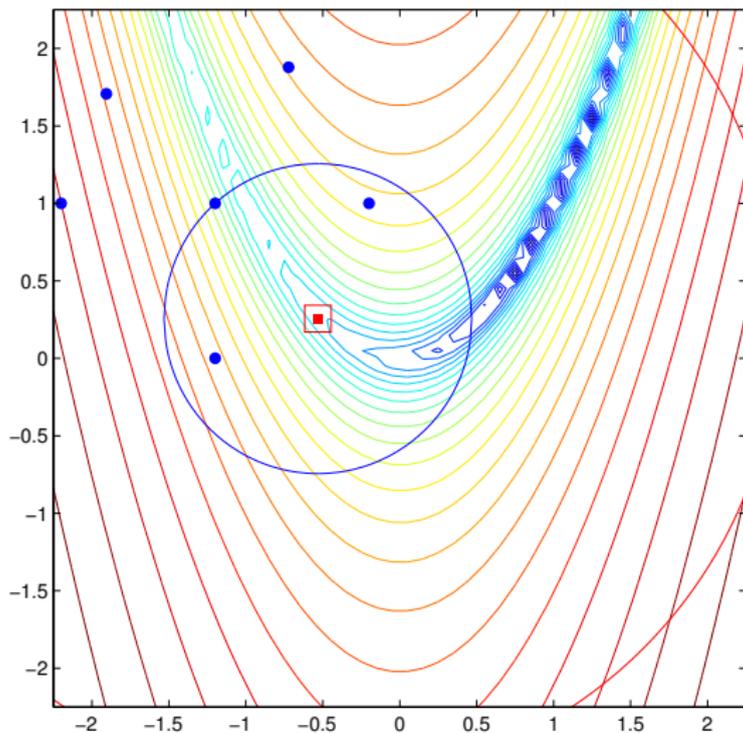
CSV - Regression



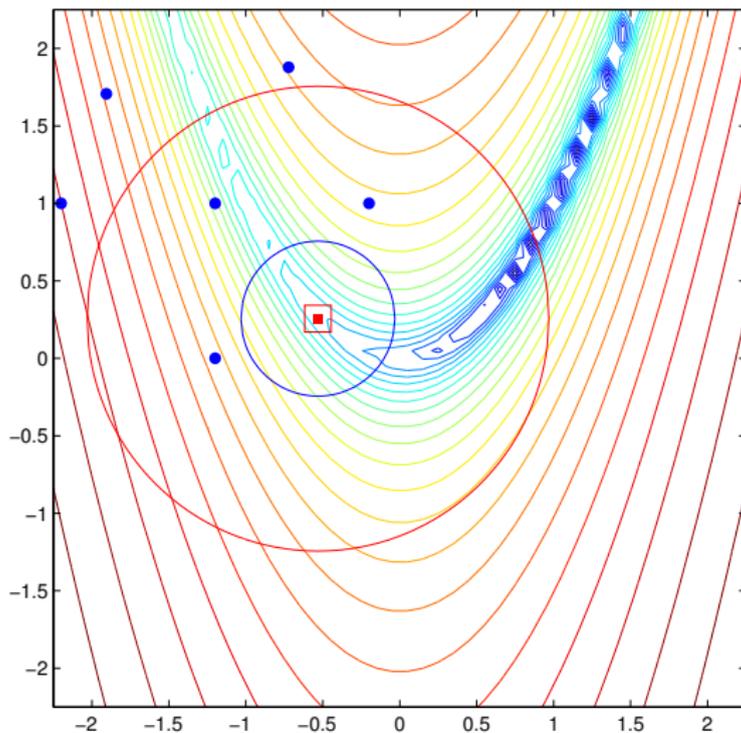
CSV - Regression



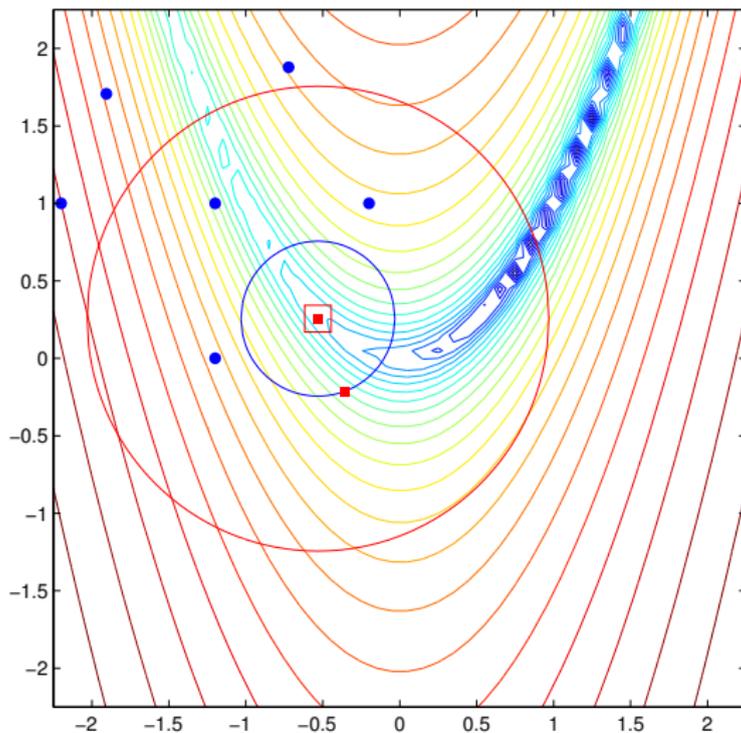
CSV - Regression



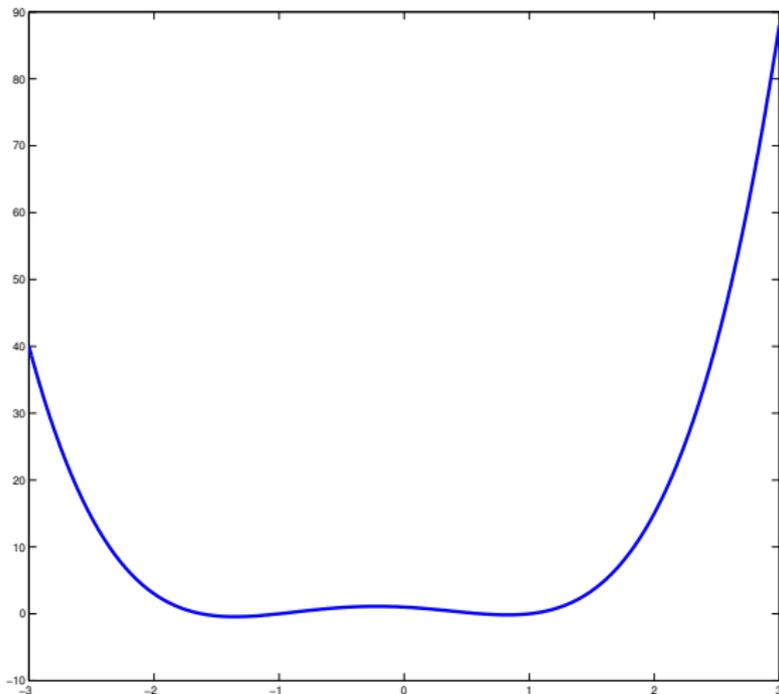
CSV - Regression



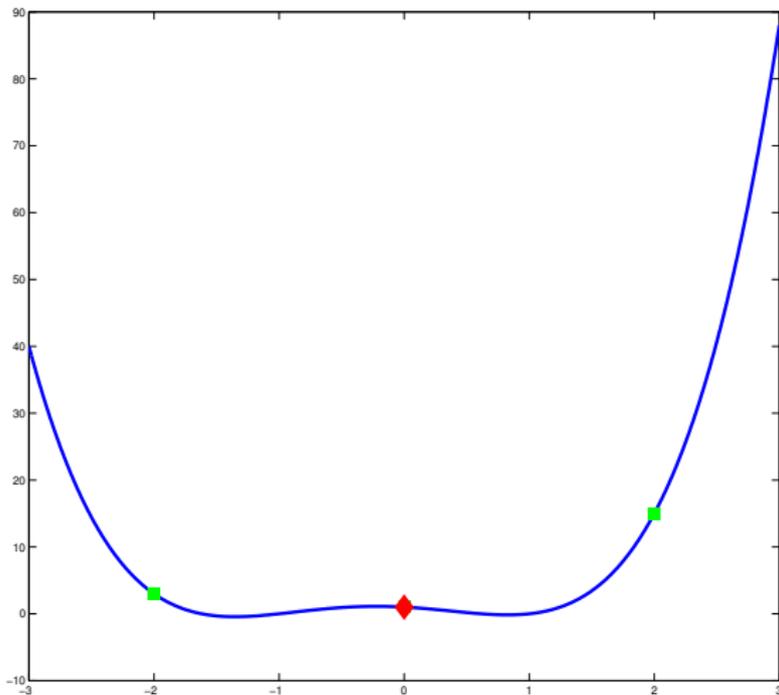
CSV - Regression



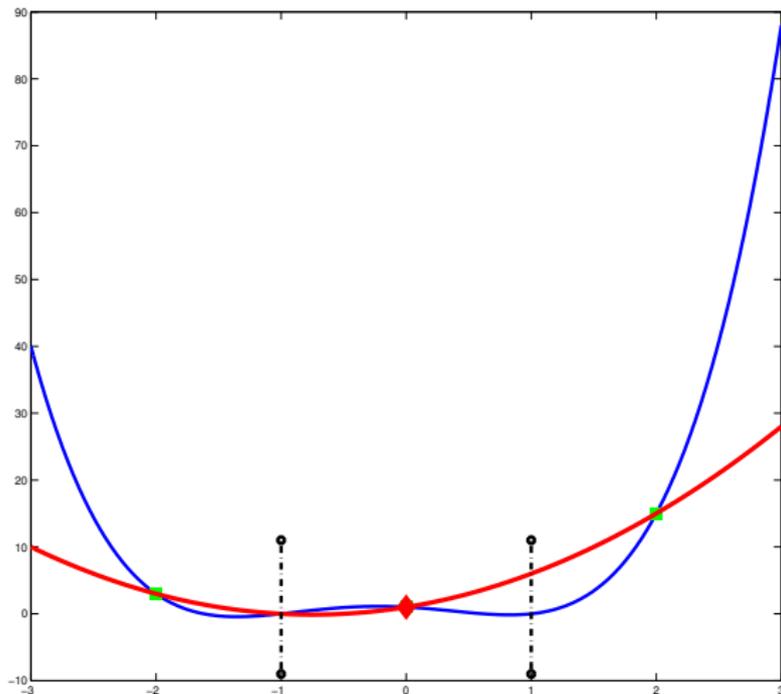
Extra Care - Can't Keep Every Point



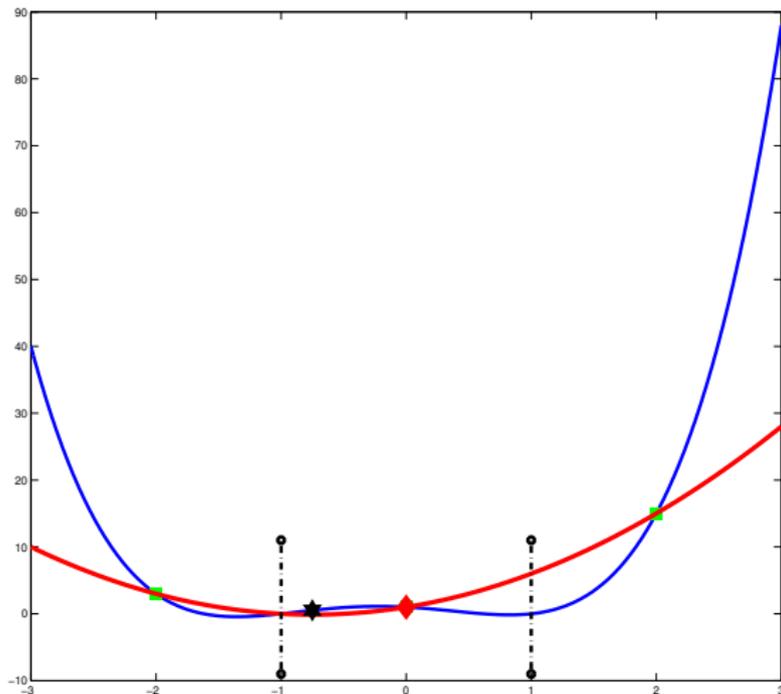
Extra Care - Can't Keep Every Point



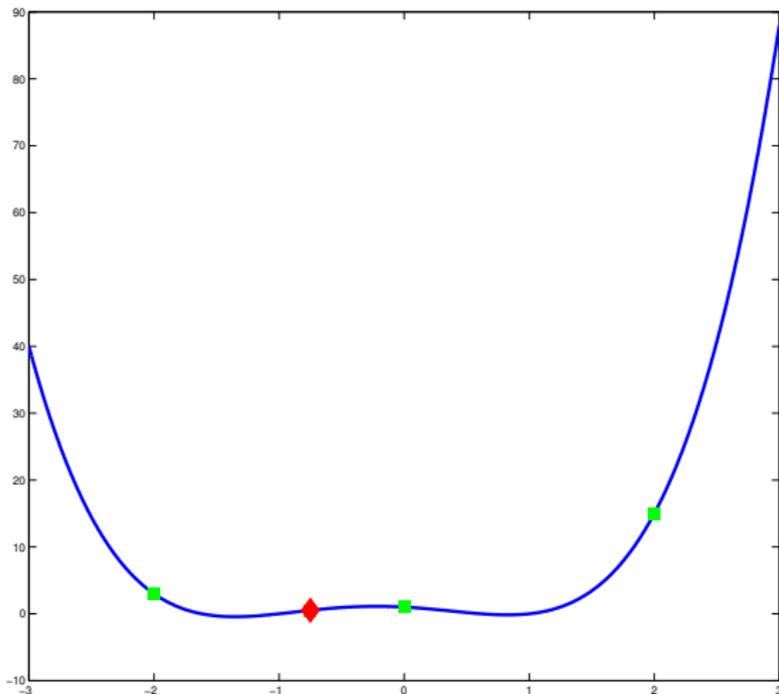
Extra Care - Can't Keep Every Point



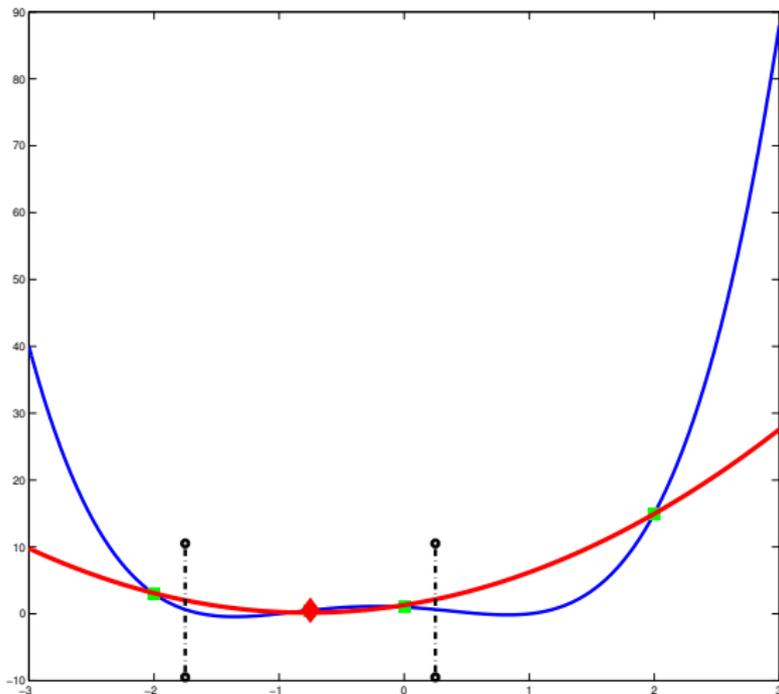
Extra Care - Can't Keep Every Point



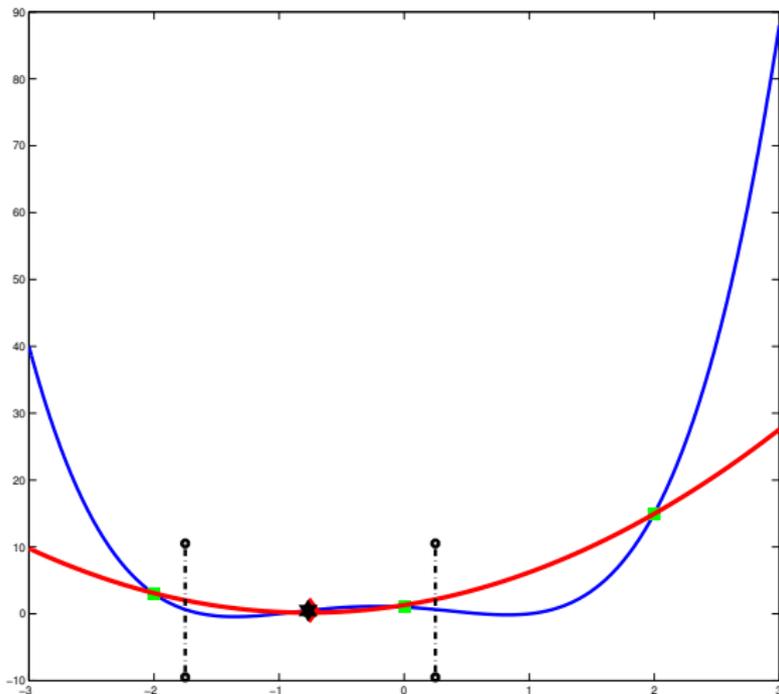
Extra Care - Can't Keep Every Point



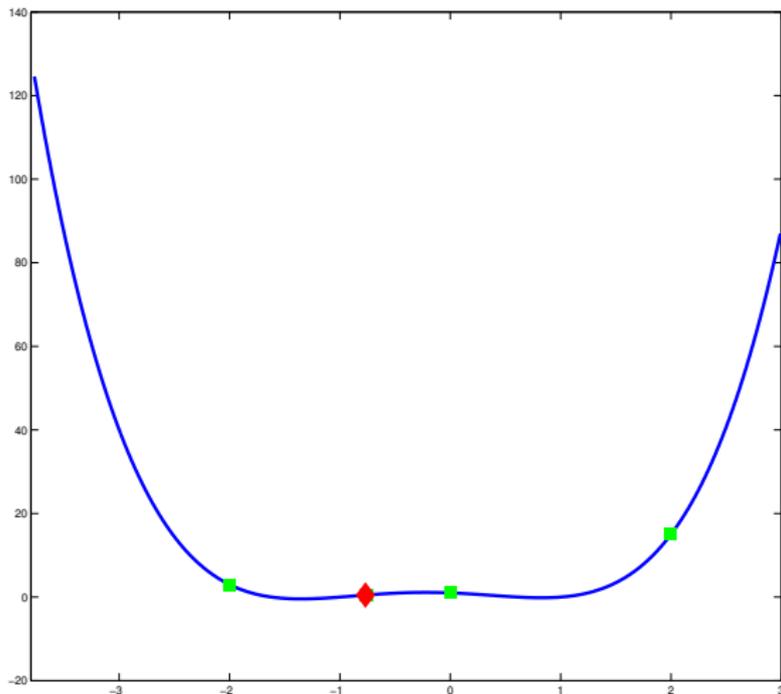
Extra Care - Can't Keep Every Point



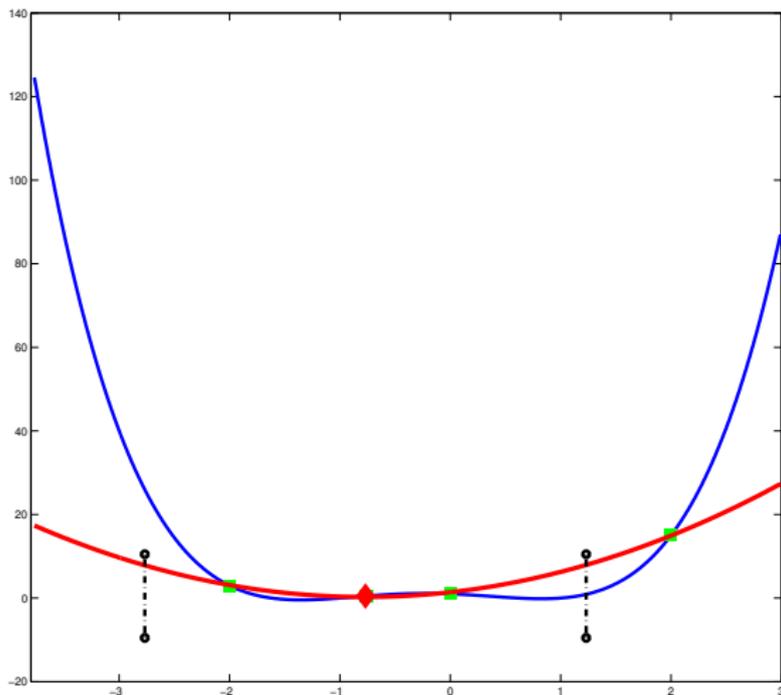
Extra Care - Can't Keep Every Point



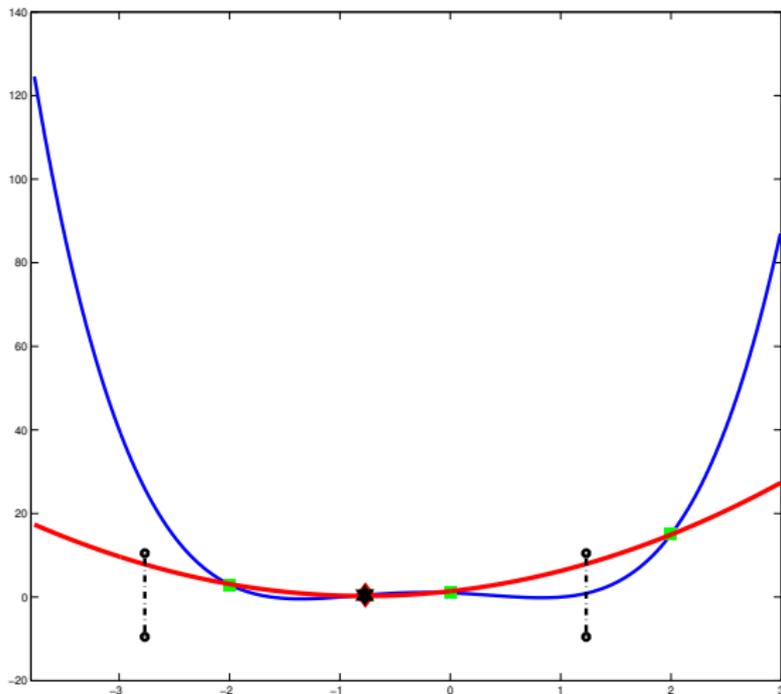
Extra Care - Can't Keep Every Point



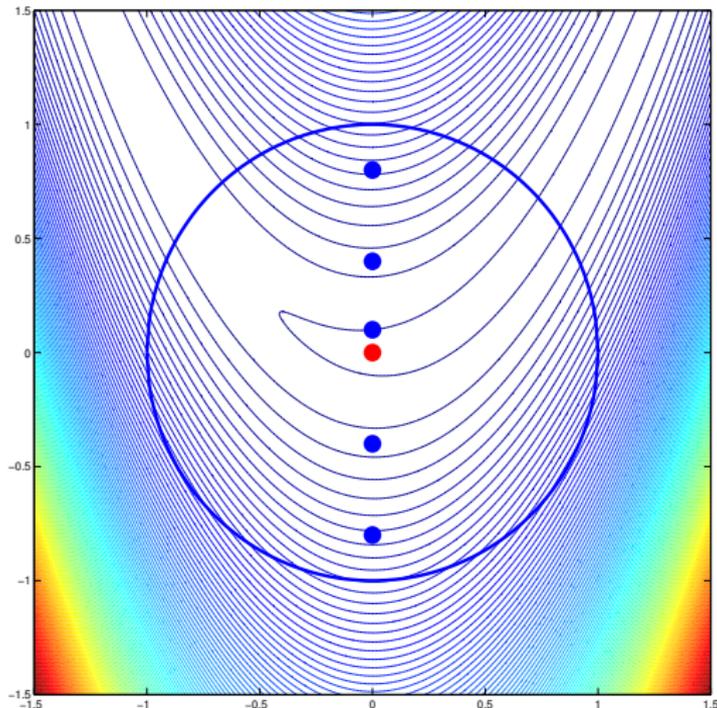
Extra Care - Can't Keep Every Point



Extra Care - Can't Keep Every Point



Extra Care - Sample Geometry



CSV Framework - Quadratic Models

Given a function f to minimize, an initial starting point x_0 , and trust-region radius Δ_0 , build a model m_0 :

- 1 Criticality Step
- 2 Compute a step s_k which minimizes m_k on $B(x_k, \Delta_k)$
- 3 Compute $f(x_k + s_k)$ and $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$
- 4 Update Δ_k and m_k

CSV Framework - Quadratic Models

Given a function f to minimize, an initial starting point x_0 , and trust-region radius Δ_0 , build a model m_0 :

- ① Criticality Step
- ② Compute a step s_k which minimizes m_k on $B(x_k, \Delta_k)$
- ③ Compute $f(x_k + s_k)$ and $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$
- ④ Update Δ_k and m_k

Our Paradigm

Assume our problem has the form:

$$\min_{x \in \mathbb{R}^n} f(x, k) = f_s(x) + \varepsilon(x, k)$$

where k significantly affects the computational time of f .

- Uncertainty varies at different points since we can choose different values of k .
- Our goal is to develop an algorithm that takes advantage of knowledge of the uncertainty.
- As a starting point, we investigate how much we can improve convergence if complete knowledge of $\varepsilon(x, k)$ is known.

Our Paradigm

Assume our problem has the form:

$$\min_{x \in \mathbb{R}^n} f(x, k) = f_s(x) + \varepsilon(x, k)$$

where k significantly affects the computational time of f .

- Uncertainty varies at different points since we can choose different values of k .
- Our goal is to develop an algorithm that takes advantage of knowledge of the uncertainty.
- As a starting point, we investigate how much we can improve convergence if complete knowledge of $\varepsilon(x, k)$ is known.

Our Paradigm

Assume our problem has the form:

$$\min_{x \in \mathbb{R}^n} f(x, k) = f_s(x) + \varepsilon(x, k)$$

where k significantly affects the computational time of f .

- Uncertainty varies at different points since we can choose different values of k .
- Our goal is to develop an algorithm that takes advantage of knowledge of the uncertainty.
- As a starting point, we investigate how much we can improve convergence if complete knowledge of $\varepsilon(x, k)$ is known.

Our Paradigm

Assume our problem has the form:

$$\min_{x \in \mathbb{R}^n} f(x, k) = f_s(x) + \varepsilon(x, k)$$

where k significantly affects the computational time of f .

- Uncertainty varies at different points since we can choose different values of k .
- Our goal is to develop an algorithm that takes advantage of knowledge of the uncertainty.
- As a starting point, we investigate how much we can improve convergence if complete knowledge of $\varepsilon(x, k)$ is known.

Our Paradigm

If we have:

- f evaluated at $\{x_0, \dots, x_p\}$ and
- $\sigma_{noise_i}^2 = \text{Var}(f(x_i))$

then the model m , which is the best linear unbiased estimator of f , is the solution to the weighted least squares problem:

$$\min \sum_{i=0}^p w_i (m(x_i) - f(x_i))^2$$

with weights $w_i \propto \frac{1}{\sigma_{total_i}^2}$.

Our Approach

Say we had the second-order Taylor expansion $m(x)$ of $f_s(x)$ at our trust region center x_0 . At a given point x_i , consider the difference $f(x_i) - m(x_i)$ to be a random process with variance $\sigma_{total_i}^2$.

Consider this variance to have two parts:

- The difference between the function and the Taylor model $\sigma_{Taylor_i}^2$
- The noise added to the smooth function $\sigma_{noise_i}^2$

Our Approach

Say we had the second-order Taylor expansion $m(x)$ of $f_s(x)$ at our trust region center x_0 . At a given point x_i , consider the difference $f(x_i) - m(x_i)$ to be a random process with variance $\sigma_{total_i}^2$.

Consider this variance to have two parts:

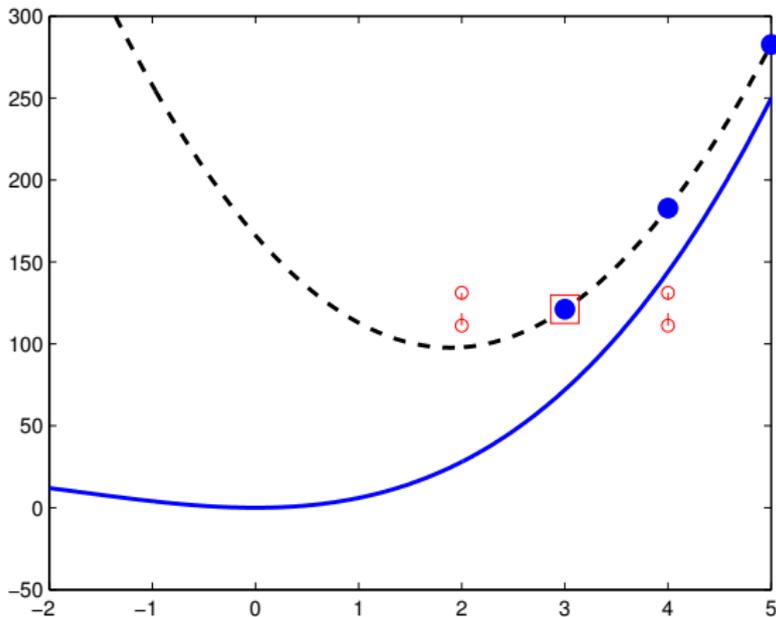
- The difference between the function and the Taylor model $\sigma_{Taylor_i}^2$
- The noise added to the smooth function $\sigma_{noise_i}^2$

Our Approach

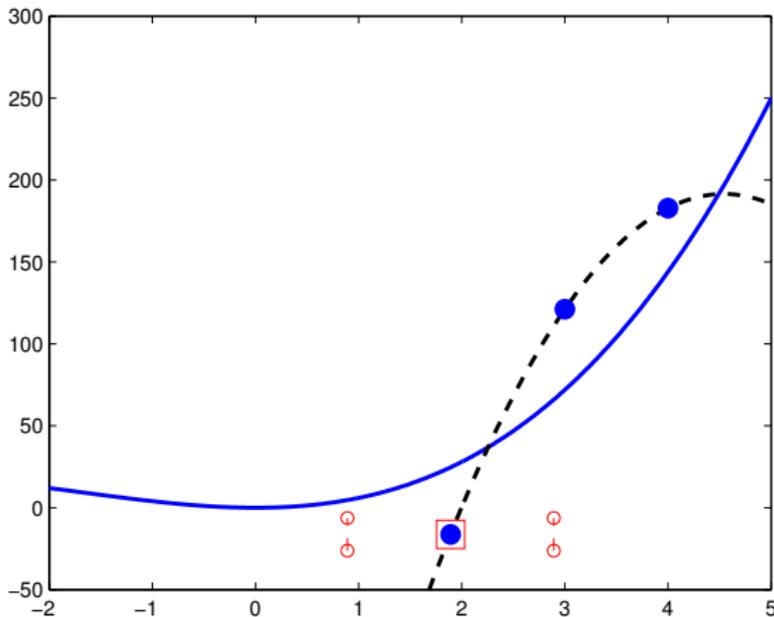
$$\begin{aligned}
 w_i &\propto \frac{1}{\sigma_{total_i}^2} \\
 &= \frac{1}{C\sigma_{noise_i}^2 + \sigma_{Taylor_i}^2} \\
 &= \frac{1}{C\sigma_{noise_i}^2 + (L\|x_i - x_0\|^3)^2}
 \end{aligned}$$

where L is the Lipschitz constant on $\nabla^2 f_s(x)$.

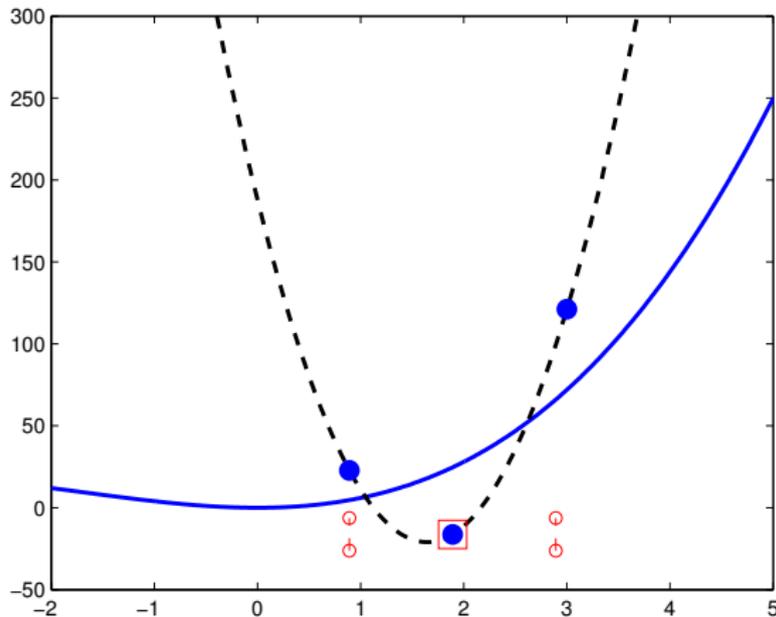
Getting stuck using ρ



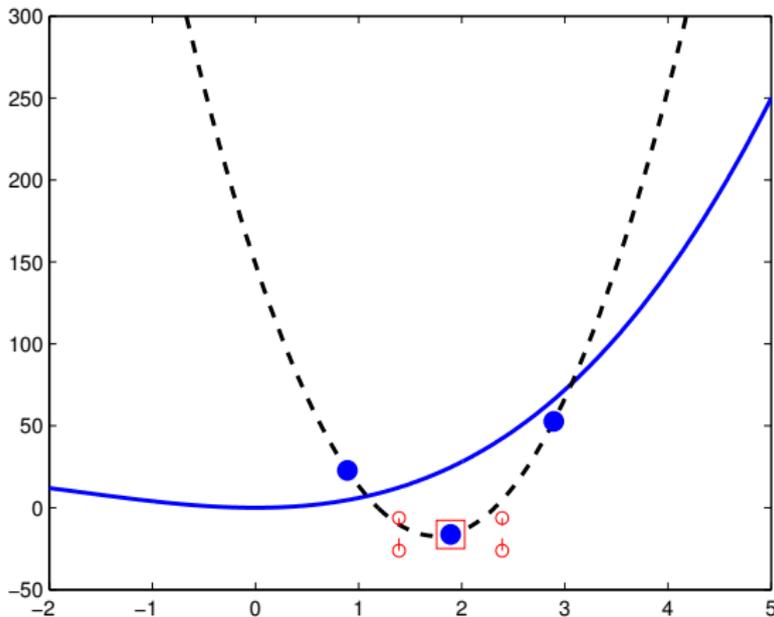
Getting stuck using ρ



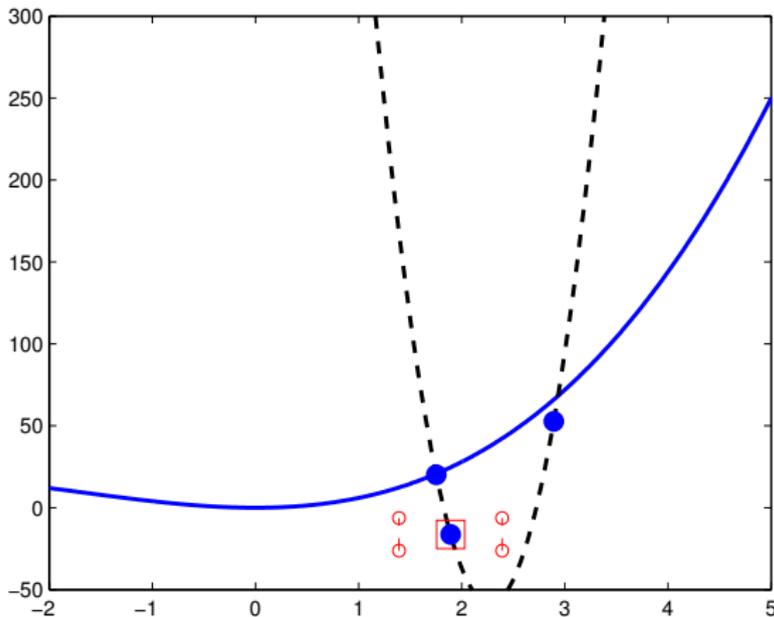
Getting stuck using ρ



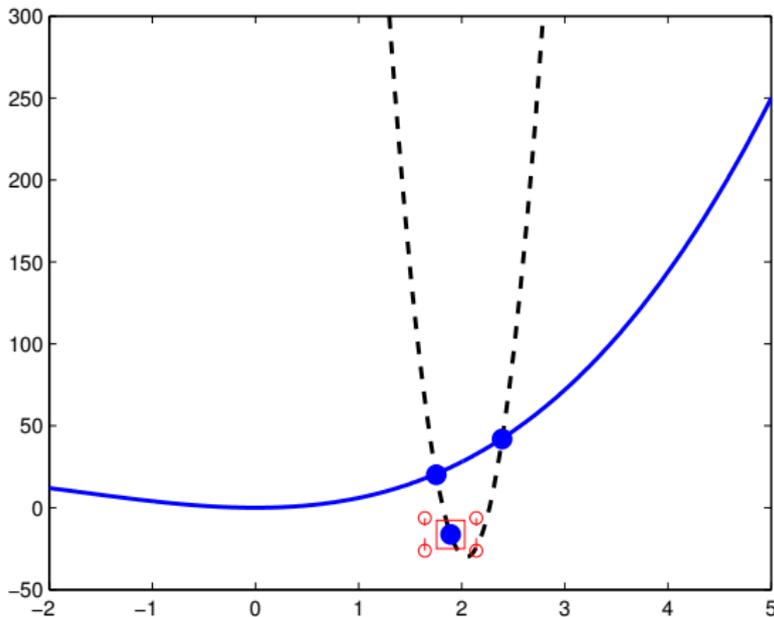
Getting stuck using ρ



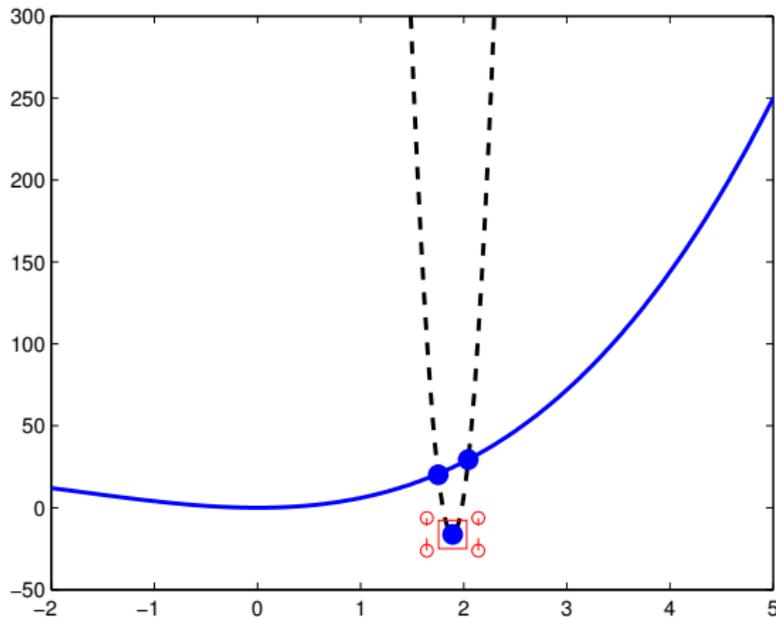
Getting stuck using ρ



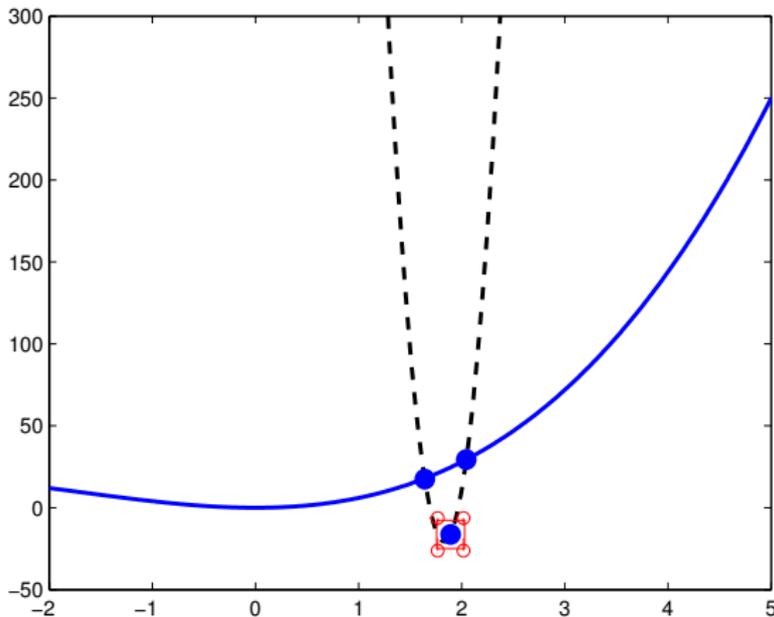
Getting stuck using ρ



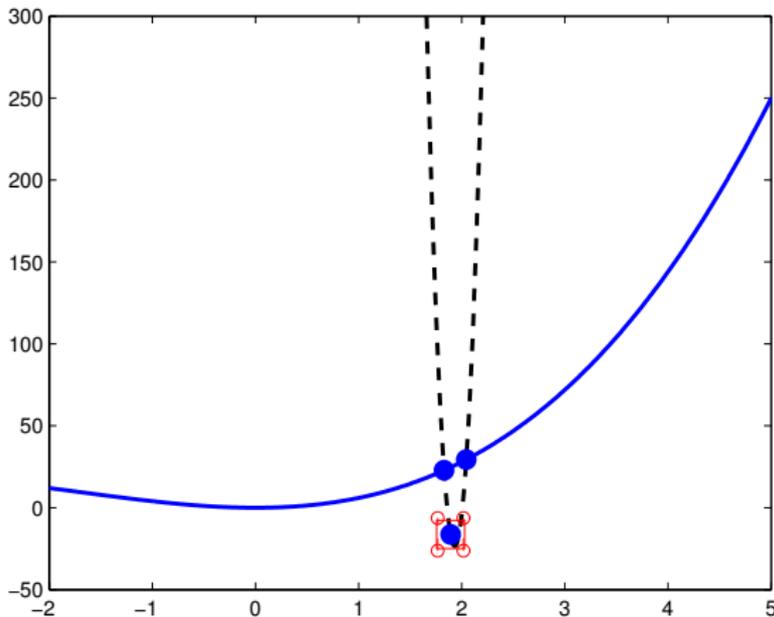
Getting stuck using ρ



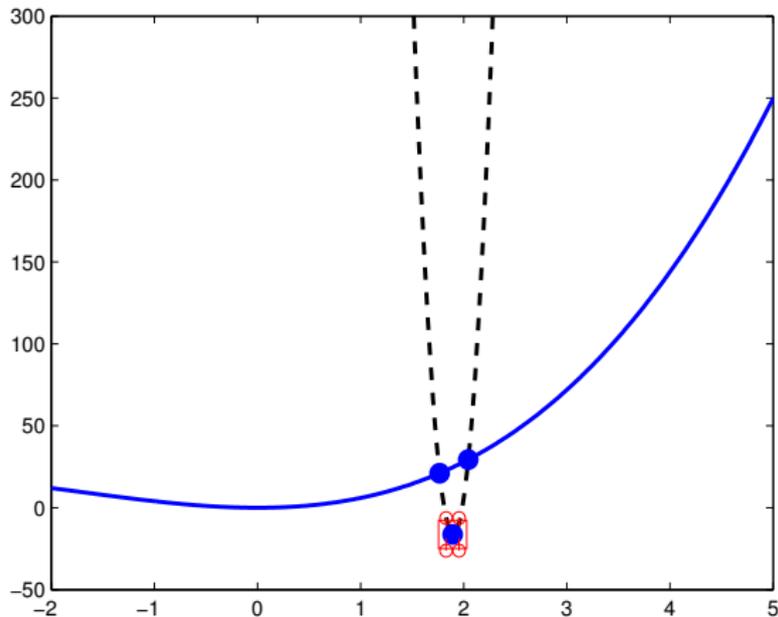
Getting stuck using ρ



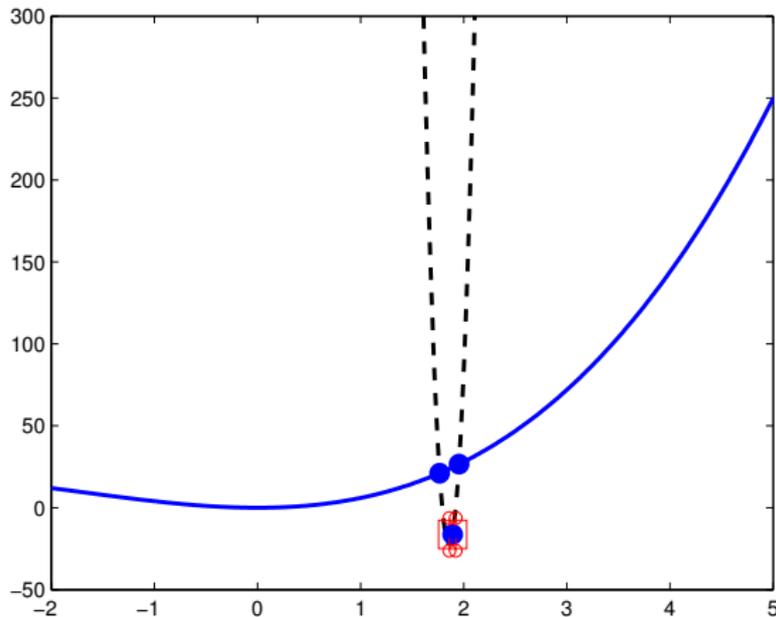
Getting stuck using ρ



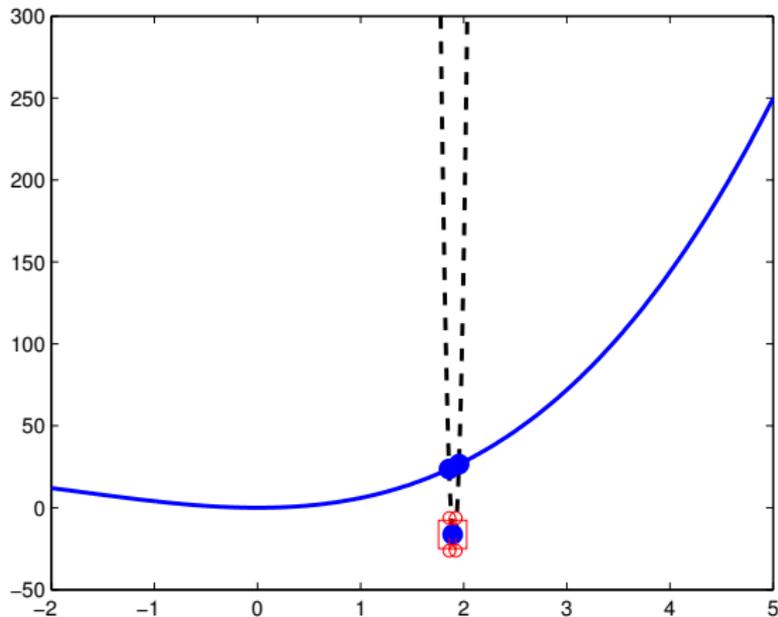
Getting stuck using ρ



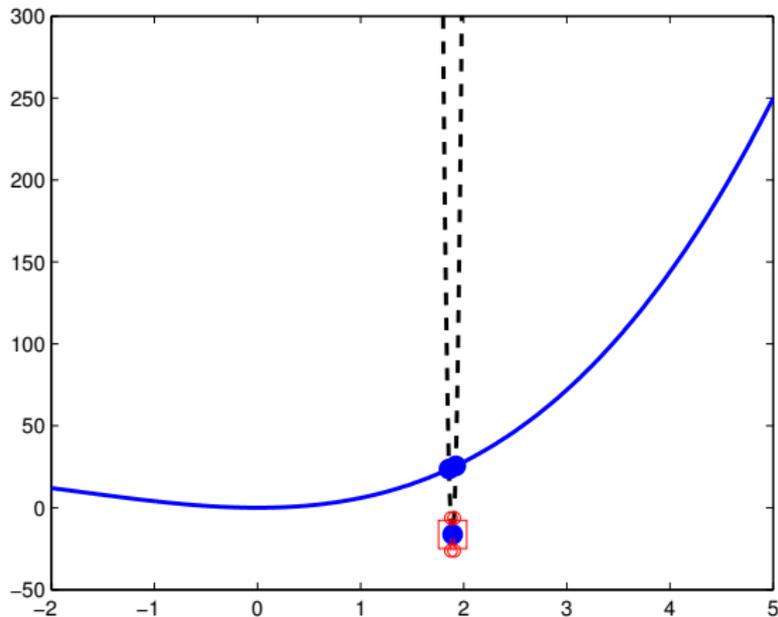
Getting stuck using ρ



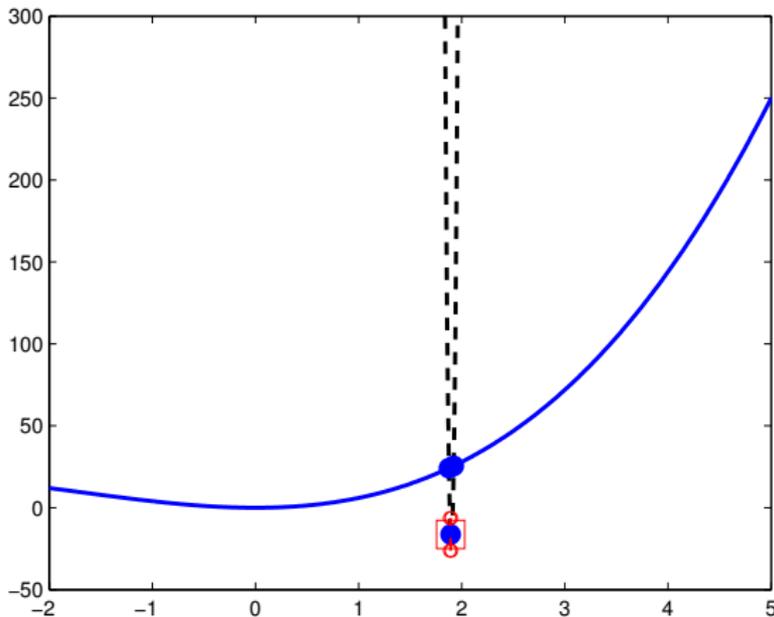
Getting stuck using ρ



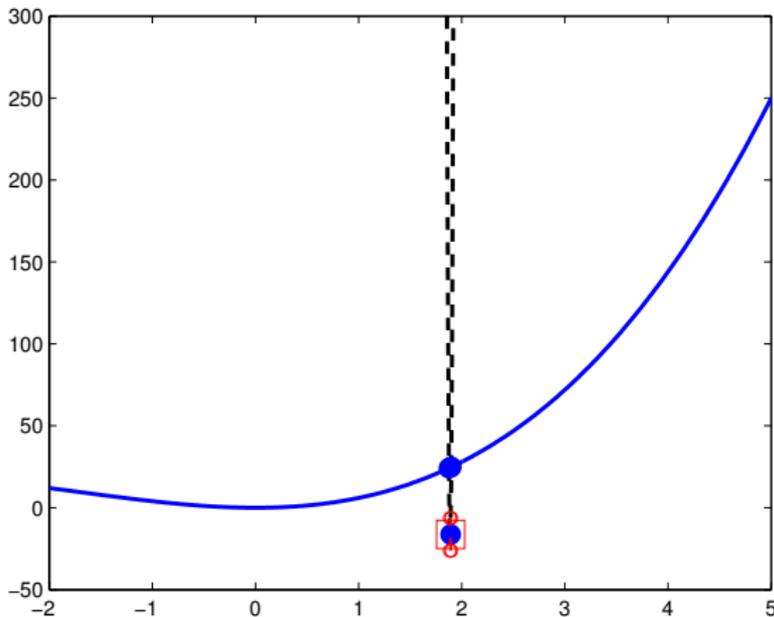
Getting stuck using ρ



Getting stuck using ρ



Getting stuck using ρ



Univariate Test Case

- Minimize problems of the form: $f(x_i) = (x_i)^3 + a(x_i)^2 + \varepsilon_i$,
 $a > 0$, $\varepsilon_i \sim \mathcal{N}(0, \Delta_{k_i}^3/1000)$

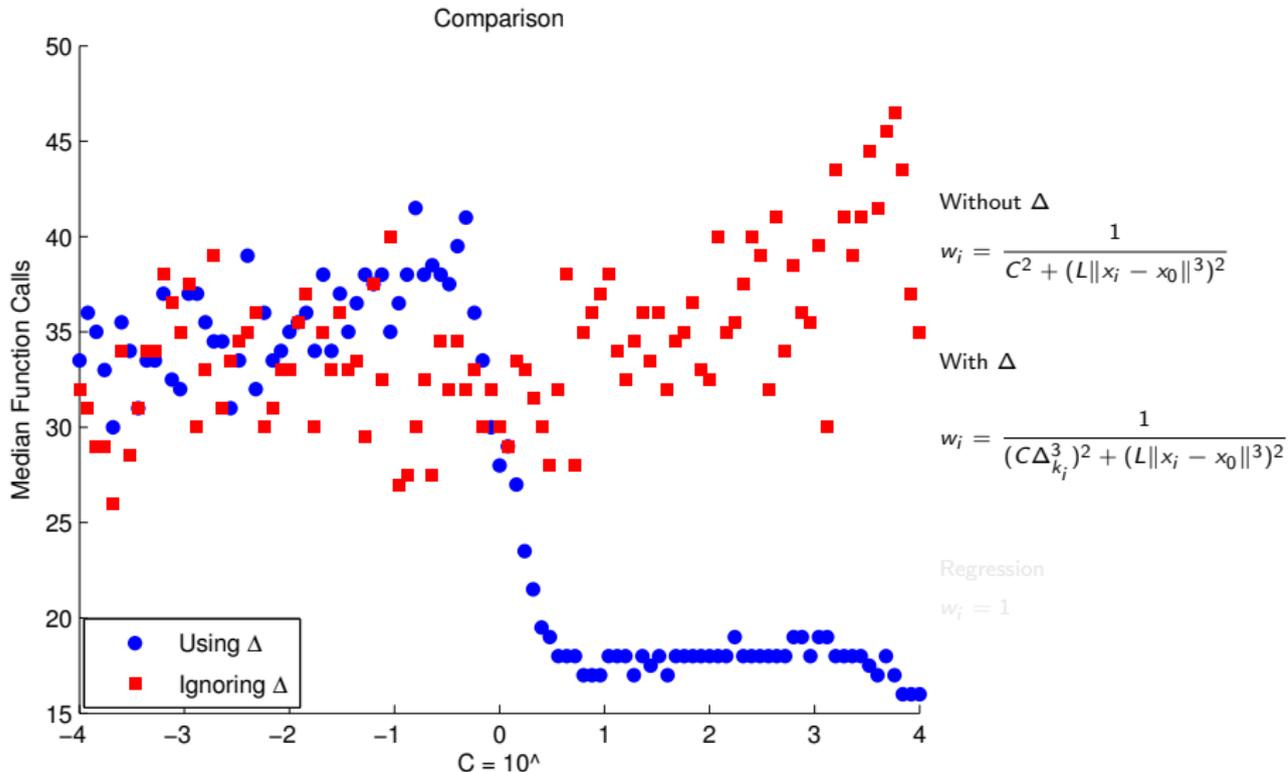
- Compare three different weighting schemes:

- $w_i = \frac{1}{C^2 + (L\|x_i - x_0\|^3)^2}$
- $w_i = \frac{1}{(C\Delta_{k_i}^3)^2 + (L\|x_i - x_0\|^3)^2}$
- $w_i = 1$

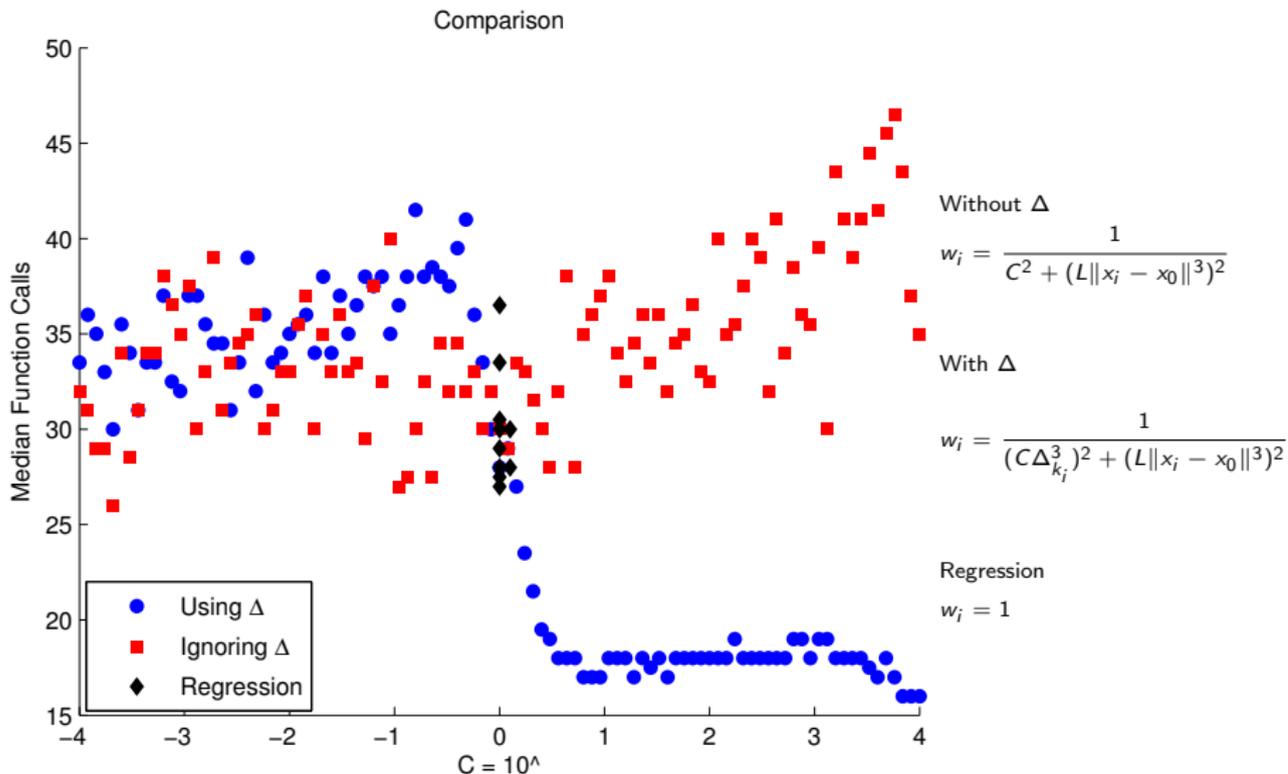
Univariate Test Case

- Minimize problems of the form: $f(x_i) = (x_i)^3 + a(x_i)^2 + \varepsilon_i$,
 $a > 0$, $\varepsilon_i \sim \mathcal{N}(0, \Delta_{k_i}^3/1000)$
- Compare three different weighting schemes:
 - $w_i = \frac{1}{C^2 + (L\|x_i - x_0\|^3)^2}$
 - $w_i = \frac{1}{(C\Delta_{k_i}^3)^2 + (L\|x_i - x_0\|^3)^2}$
 - $w_i = 1$

Median Number of Iterates

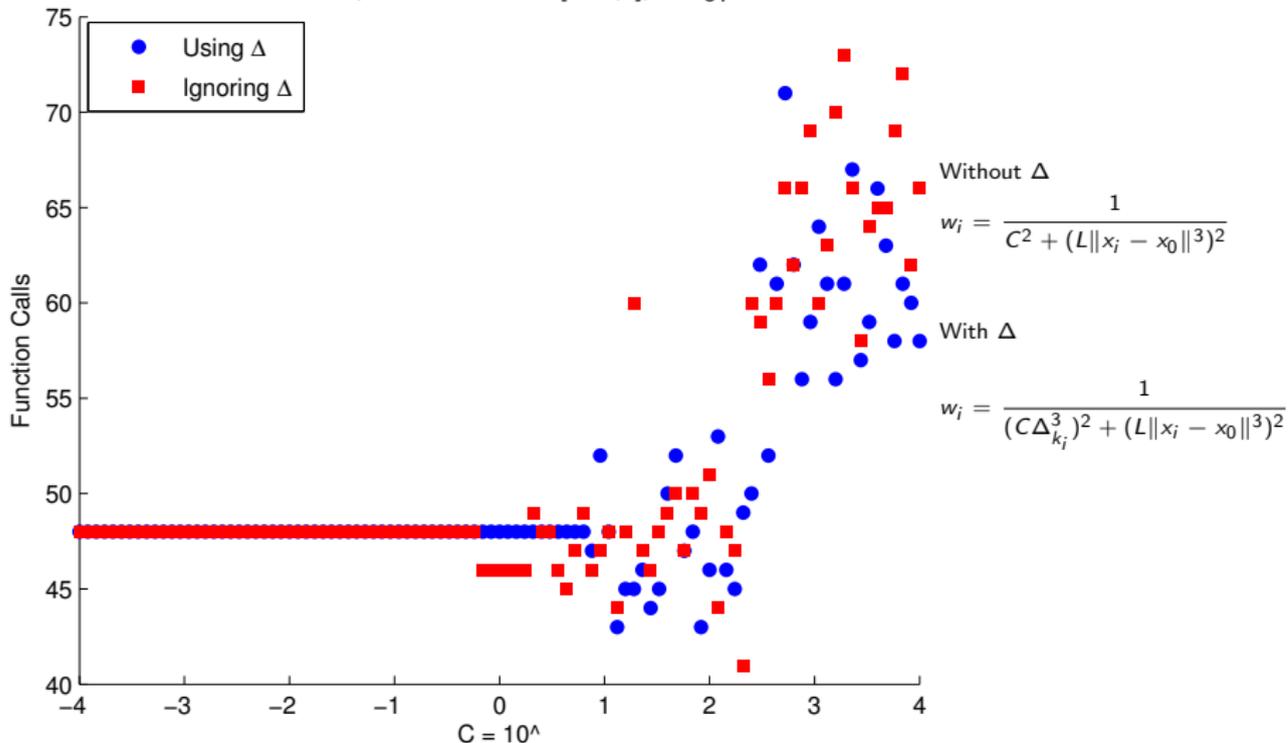


Median Number of Iterates



Rosenbrock - Deterministic Noise

$\sigma = 10^{-3}$; Rosenbrock from $[-1.2; 1]$; Using p



Higher Dimensions

- For 22 nonlinear least squares functions from the CUTEr collection.
- Each problem has the form:

$$f(x) = (1 + \varepsilon_i \phi(x)) \sum_{k=1}^p g_k(x)^2$$

- ε is the relative noise level proportional to $\Delta_{k_i}^3$
- $\phi(x)$ is a function which oscillates between -1 and 1 with high-frequency and low-frequency oscillations.
- With different starting values and values for p , we end up with 53 benchmark problems.

Higher Dimensions

- For 22 nonlinear least squares functions from the CUTer collection.
- Each problem has the form:

$$f(x) = (1 + \varepsilon_i \phi(x)) \sum_{k=1}^p g_k(x)^2$$

- ε is the relative noise level proportional to $\Delta_{k_i}^3$
- $\phi(x)$ is a function which oscillates between -1 and 1 with high-frequency and low-frequency oscillations.
- With different starting values and values for p , we end up with 53 benchmark problems.

Higher Dimensions

- For 22 nonlinear least squares functions from the CUTEr collection.
- Each problem has the form:

$$f(x) = (1 + \varepsilon_i \phi(x)) \sum_{k=1}^p g_k(x)^2$$

- ε is the relative noise level proportional to $\Delta_{k_i}^3$
 - $\phi(x)$ is a function which oscillates between -1 and 1 with high-frequency and low-frequency oscillations.
- With different starting values and values for p , we end up with 53 benchmark problems.

Higher Dimensions

- Let f_i^* be the smallest value found by any solver in 1500 iterates.
- Let f_i^0 be the initial value for each solver.

Consider each solved when it finds a function value below

$$f_i^* + 10^{-3}(f_i^0 - f_i^*)$$

	Solves first	Solves eventually
<i>Interpolation</i>	35%	77%
<i>Regression</i>	33%	77%
<i>Weighted</i> $_{\Delta, C=10^{-2}}$	57%	80%

Higher Dimensions

- Let f_i^* be the smallest value found by any solver in 1500 iterates.
- Let f_i^0 be the initial value for each solver.

Consider each solved when it finds a function value below

$$f_i^* + 10^{-3}(f_i^0 - f_i^*)$$

	Solves first	Solves eventually
<i>Interpolation</i>	35%	77%
<i>Regression</i>	33%	77%
<i>Weighted</i> $_{\Delta, C=10^{-2}}$	57%	80%

Summary and Questions

- We demonstrated a weighting scheme which can improve on performance of the CSV algorithm when we have complete knowledge about the uncertainty and the Lipschitz constant.
- We are hopeful that we can devise an adaptive scheme when we don't have complete knowledge.
- We believe this technique can be modified for different measures of uncertainty quantification.
- What termination criteria makes sense for noisy problems?